Agus Kurniawan

Intelligent IoT Projects in 7 Days

Build exciting projects using smart devices

Packt

Intelligent IoT Projects in 7 Days

Build exciting projects using smart devices

Agus Kurniawan

Packt

BIRMINGHAM - MUMBAI

Intelligent IoT Projects in 7 Days

First published: September 2017

Production reference: 1070917

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham

B3 2PB, UK.

ISBN 978-1-78728-642-9

www.packtpub.com

Credits

Author Agus Kurniawan	**Copy Editor** Stuti Srivastava
Reviewers Ruben Oliva Ramos Pradeeka Seneviratne Vasilis Tzivaras	**Project Coordinator** Virginia Dias
Commissioning Editor Kartikey Pandey	**Proofreader** Safis Editing
Acquisition Editor Prateek Bharadwaj	**Indexer** Aishwarya Gangawane
Content Development Editor Sharon Raj	**Graphics** Kirk D'Penha
Technical Editor Mohit Hassija	**Production Coordinator** Aparna Bhagat

About the Author

Agus Kurniawan is a lecturer, IT consultant, and an author. He has experience in various software and hardware development projects, delivering materials in training and workshops, and delivering technical writing for 17 years. He has been awarded the Microsoft Most Valuable Professional (MVP) award for 13 years in a row.

He is currently doing some research and teaching activities related to networking and security systems at the Faculty of Computer Science, University of Indonesia, and the Samsung R&D Institute, Indonesia. Currently, he's pursuing a PhD in Computer Science in Germany.

I am thankful to the entire team at Packt, especially Prateek Bharadwaj, Sharon Raj, Mohit Hassija, and the reviewers, for being so cooperative and patient with me. They have been a great help and their feedback and tips have improved the manuscript considerably.
This book is dedicated to all open source makers, developers, contributors, and enthusiasts of Arduino, Raspberry Pi, and Machine Learning communities. Thanks to Makers.ID and the SeeedStudio community for the hardware support.

I would like to thank my wife, Ela Juitasari; my son, Thariq; and my daughter, Zahra; for their support and encouragement.

About the Reviewers

Ruben Oliva Ramos is a Computer Systems Engineer from Tecnologico of León Institute, with a master's degree in Computer and Electronic Systems Engineering, Teleinformatics and Networking Specialization from the University of Salle Bajio in Leon, Guanajuato Mexico. He has more than five years of experience in developing WEB applications to control and monitor devices connected to Arduino, and Raspberry Pi using WEB Frameworks and Cloud Services to build the Internet of Things applications.

He is a mechatronics teacher the University of Salle Bajio and teaches students in the master's degree in Design and Engineering of Mechatronics Systems, he also works at Centro de Bachillerato Tecnologico Industrial 225 in Leon, Guanajuato Mexico, teaching subjects like: Electronics, Robotics and Control, Automation and Microcontrollers at the Mechatronics Technician Career, Consultant, and developer projects in areas like: Monitoring systems and data logger data using technologies: Android, iOS, Windows Phone, HTML5, PHP, CSS, Ajax, JavaScript, Angular, ASP .NET databases: SQLite, MongoDB, MySQL, WEB Servers: Node.js, IIS, hardware programming: Arduino, Raspberry Pi, Ethernet Shield, GPS and GSM/GPRS, ESP8266, control and monitor Systems for data Acquisition and Programming.

He has written the book *Internet of Things Programming with JavaScript*, by Packt. Also *Monitoring, Controlling and Acquisition of Data with Arduino and Visual Basic .NET* for Alfaomega.

I would like to thank my Savior and Lord, Jesus Christ for giving me strength and courage to pursue this project; to my dearest wife, Mayte, our two lovely sons, Ruben and Dario, To my father (Ruben), my dearest mom (Rosalia), my brother (Juan Tomas), and my sister (Rosalia) whom I love, for all their support while reviewing this book, for allowing me to pursue my dream and tolerating my not being with them after my busy day job.

I'm very grateful to Packt Publishing for giving the opportunity to collaborate as an author and reviewer, and to be a part of this honest and professional team.

Pradeeka Seneviratne is a software engineer with over 10 years of experience in computer programming and systems designing. He is an expert in the development of Arduino and Raspberry Pi-based embedded systems. Pradeeka is currently a full-time embedded software engineer who works with embedded systems and highly scalable technologies. Previously, he worked as a software engineer for several IT infrastructures and technology servicing companies. He collaborated with the Outernet (Free data from space, forever) project as a hardware volunteer and a software tester for Lighthouse, and Raspberry Pi-based DIY Outernet receivers based on Ku band satellite frequencies. He is also the author of four books:

- *Internet of Things with Arduino Blueprints*, by Packt
- *IoT: Building Arduino-based Projects*, by Packt
- *Building Arduino PLCs*, by Apress
- *Raspberry Pi 3 Projects for Java Programmers*, by Packt

Vasilis Tzivaras is a Computer Engineer holding a BSc degree in Computer Science and Engineering, University of Ioannina in Greece. His thesis is about the autonomous landing of a quadcopter using IBVS. During his studies, he was the chair of the IEEE Student Branch of the University of Ioannina. With his colleagues, he has organized workshops and lead projects relevant to robotics and other technologies. Using Arduino, Raspberry Pi, and other low cost and energy solutions he has taken part in many hackathons and contests and has also taken a good place. He is the writer of *Building a Quadcopter with Arduino* that was published in 2015 and *Raspberry Pi Zero W Wireless Projects* that will be published by the end of 2017.

Get the most in-demand software skills with Mapt. Mapt gives you full access to all Packt books and video courses, as well as industry-leading tools to help you plan your personal development and advance your career.

Customer Feedback

Table of Contents

Preface

Internet of Things (IoT) is a ground-breaking technology that involves connecting numerous physical devices to the Internet and controlling them. Analyzing data from Internet of Things devices and converting it into something meaningful is currently driving the next level of IoT learning. Discover how to build your own Intelligent Internet of Things projects and bring a new degree of interconnectivity to your world.

What this book covers

Chapter 1, *A Simple Smart Gardening System*, begins with explaining how to build a simple smart gardening system with involved plant sensor devices and Arduino.

Chapter 2, *A Smart Parking System*, will teach you to build a smart parking system. Learn how to detect a car plate and to count the car parking duration. Various pattern recognition algorithms will be introduced to detect a car plate.

Chapter 3, *Making Your Own Vending Machine*, will help you build a simple vending machine, detecting a coin is an important part of the vending machine and building UI (User Interface) for the vending machine is explored.

Chapter 4, *A Smart Digital Advertising Dashboard*, teaches you to build a simple smart digital advertising which could detect people's presence so the advertiser can obtain an effective report of how many people watch the ads display.

Chapter 5, *A Smart Speaker Machine*, will help you build a simple smart speaker machine. You will start to learn Amazon Echo and then build your own simple smart speaker machine.

Chapter 6, *Autonomous Firefighter Robot*, teaches you to build an autonomous robot which finds a fire source location and extinguish fires. You will also learn to find the fire source location and navigate to the source location.

Chapter 7, *Multi-Robot Cooperation Using Swarm Intelligence*, focuses on how to make transport cooperation among robots using swarm intelligence. You will also learn the concept of swarm intelligence so that you can implement it in among robots.

Appendix, *Essential Hardware Components*, covers mandatory hardware required for this book.

What you need for this book

In order to work with Drupal 8 and to run the code examples found in this book, the following software will be required.

Web server software stack:

- Web server: Apache (recommended), Nginx, or Microsoft IIS
- Database: MySQL 5.5 or MariaDB 5.5.20 or higher
- PHP: PHP 5.5.9 or higher

Chapter 1, *A Simple Smart Gardening System*, details all of these requirements and includes a recipe highlighting an out of the box development server setup.

You will also need a text editor, following is a suggestion of popular editors and IDEs:

- Atom.io editor, https://atom.io/
- PHPStorm (specific Drupal integration), https://www.jetbrains.com/phpstorm/
- Vim with Drupal configuration, https://www.drupal.org/project/vimrc

Your operating system's default text editor or command line file editors For more information on hardware retirements, you may refer *Appendix* section of this book

Who this book is for

If you're a developer, IoT enthusiast, or just someone curious about Internet of Things, then this book is for you. A basic understanding of electronic hardware, networking, and basic programming skills would do wonders.

Conventions

In this book, you will find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning. Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "In the `loop()` function, we read the value of the distance from the sensor and then display it on the serial port."

A block of code is set as follows:

```
void loop() {
  int val;
  val = analogRead(A0);

  Serial.print("Soil moisture: ");
  Serial.print(val);

  delay(3000);
}
```

Any command-line input or output is written as follows:

```
$ sudo apt-get update
 $ sudo raspi-config
```

New terms and **important words** are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "Go to Sketch | Include Library | Manage Libraries and you will get a dialog."

Warnings or important notes appear like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book-what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of. To send us general feedback, simply email `feedback@packtpub.com`, and mention the book's title in the subject of your message. If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for this book from your account at http://www.packtpub.com. If you purchased this book elsewhere, you can visit http://www.packtpub.com/support and register to have the files emailed directly to you. You can download the code files by following these steps:

1. Log in or register to our website using your email address and password.
2. Hover the mouse pointer on the SUPPORT tab at the top.
3. Click on Code Downloads & Errata.
4. Enter the name of the book in the Search box.
5. Select the book for which you're looking to download the code files.
6. Choose from the drop-down menu where you purchased this book from.
7. Click on Code Download.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR / 7-Zip for Windows
- Zipeg / iZip / UnRarX for Mac
- 7-Zip / PeaZip for Linux

The code bundle for the book is also hosted on GitHub at https://github.com/PacktPublishing/Intelligent-IoT-Projects-in-7-Days. We also have other code bundles from our rich catalog of books and videos available at https://github.com/PacktPublishing/. Check them out!

Downloading the color images of this book

We also provide you with a PDF file that has color images of the screenshots/diagrams used in this book. The color images will help you better understand the changes in the output. You can download this file from https://www.packtpub.com/sites/default/files/downloads/IntelligentIoTProjectsin7Days_ColorImages.pdf.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books-maybe a mistake in the text or the code-we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting http://www.packtpub.com/submit-errata, selecting your book, clicking on the Errata Submission Form link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title. To view the previously submitted errata, go to https://www.packtpub.com/books/content/support and enter the name of the book in the search field. The required information will appear under the Errata section.

Questions

If you have a problem with any aspect of this book, you can contact us at `questions@packtpub.com`, and we will do our best to address the problem.

A Simple Smart Gardening System

Gardening is a nice activity. It needs care to keep a crop growing well. It is not possible to monitor and tend to a garden 24 hours a day, so we need a smart gardening system that can monitor and tend to the garden as we want it. This chapter will help you explore existing gardening systems and build your own simple smart gardening system.

In this chapter, we will cover the following topics:

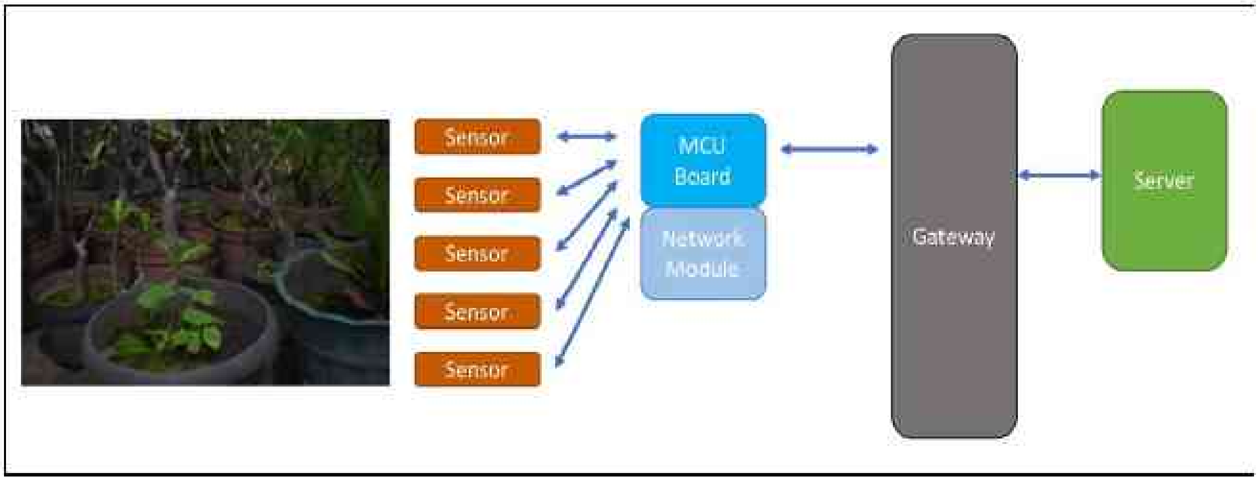
- Introducing smart gardening systems
- Exploring gardening system platforms
- Watering your garden and farm
- Sensor devices for a smart gardening system
- Watering your garden and farm
- Building a smart gardening system

Let's get started!

Introducing smart gardening system

A gardening system is a system used to practice growing and cultivating plants as a part of horticulture. A gardening system is usually developed and implemented as a manual operation. An automated gardening system is designed to enable us to manage gardening, including monitoring moisture, temperature, and humidity.

In general, we can build a smart gardening system based on the high-level architecture that is shown in the following figure:



The following is a list of components to build a smart gardening system:

- **Sensors:** Corresponding to your case, you need sensor devices to measure the garden's environment and condition. Capturing physical objects to digital form enables us to perform computing and processing.
- **MCU board with network module:** The goal of MCU is to process all data that is acquired from sensors. Most MCU boards have limited computation, so we need to send sensor data to a server for further computation. To enable us to send data, the MCU board should be attached to a network module, either Ethernet or wireless.
- **Gateway:** This is optional since some MCU boards can communicate with a server directly without modifying the protocol format. If a network module has the capability to deliver data over a primitive protocol, the

functioning of gateway is necessary because a gateway can translate one protocol format to another protocol format.

- **Server:** This is a center computation. Most servers have high-end hardware so heavy computation can be addressed.

This architecture is an ideal condition. In a real project, you may integrate an MCU board and server in one box. For instance, you can use a Raspberry Pi or BeagleBoard. These are a mini computers that you can deploy libraries to with respect to your case.

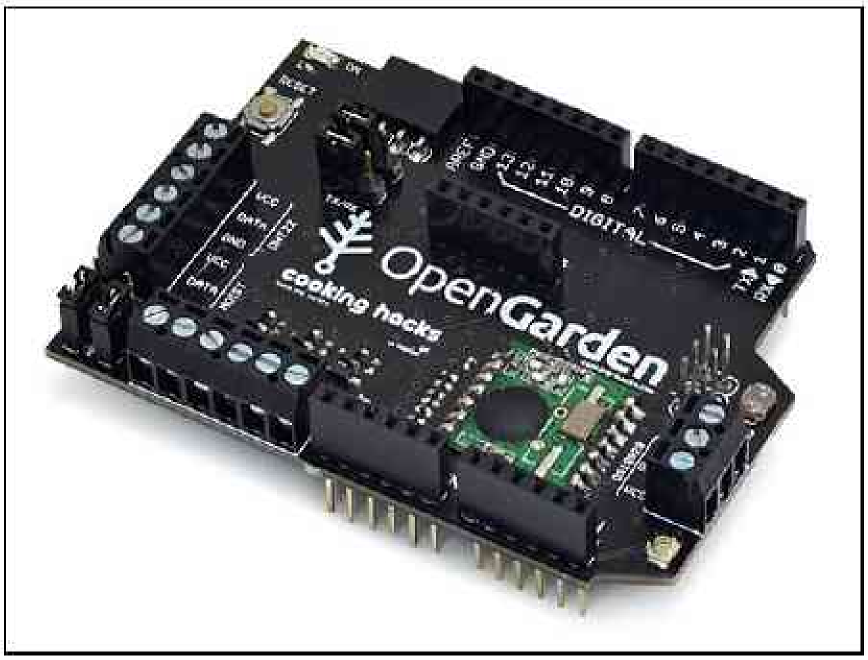
Exploring gardening system platforms

In this section, we will explore some gardening systems that could be used with our board, such as Arduino and Raspberry Pi. Some manufacturers provide kits that you can use directly in your project.

We will review three gardening system platforms that you may fit with your case.

Open Garden shield for Arduino

Open Garden shield for Arduino is manufactured by Cooking Hacks. This shield provides I/O connectors for gardening and farming sensors. It also includes an RF module with 433 MHz frequency. For further information about this shield, you can visit the official website at https://www.cooking-hacks.com/open-garden-shield-for-arduino. You can see this shield here:



Cooking Hacks provides a complete kit that you can use with Arduino directly. Currently, there are three kits. Each kit has unique features, including sensors and tools. The following is a list of Open Garden kits:

- **Indoor kit**: http://www.cooking-hacks.com/open-garden-indoor-1node-1gw
- **Outdoor kit**: http://www.cooking-hacks.com/open-garden-outdoor-1node-1gw
- **Hydroponics kit**: http://www.cooking-hacks.com/open-garden-hydroponics

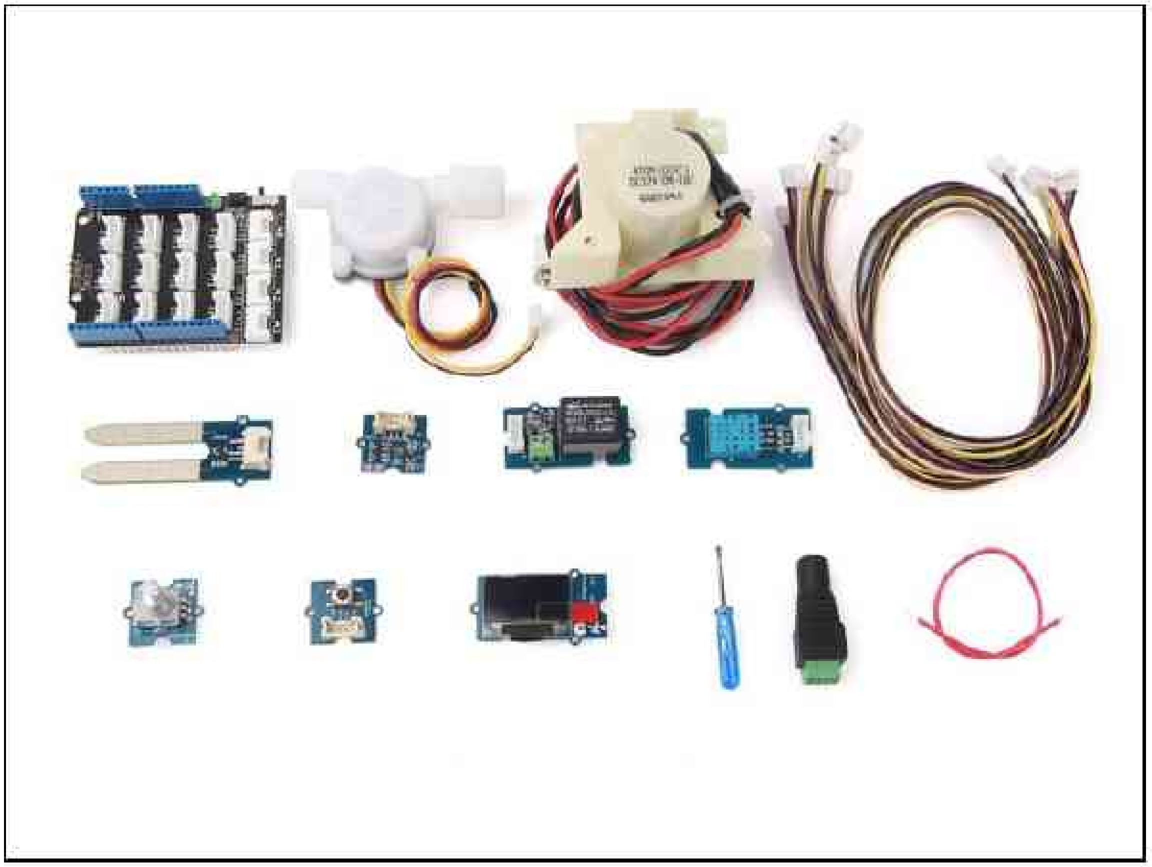
You can see a sample Open Garden kit for indoor use here:

Libelium
Powering the IoT Revolution

Grove Smart Plant Care kit for Arduino

SeeedStudio has popular products called Grove. It makes it easier for you to connect your board to various sensor and actuator devices. Currently, SeeedStudio provides the Grove Smart Plant kit for Arduino. This kit consists of temperature, humidity, soil moisture, and illumination intensity sensors. To connect these sensors, the kit provides a Grove Base shield that you can attach to Arduino.

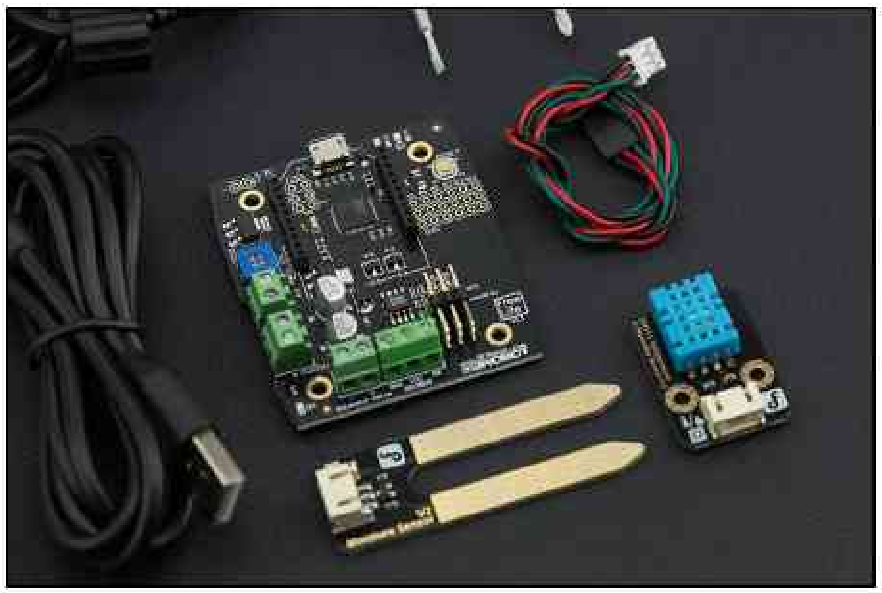
You can buy this kit on the official website, https://www.seeedstudio.com/Grove-Smart-Plant-Care-Kit-for-Arduino-p-2528.html. One type of Grove smart plant care kit for Arduino is shown here:



EcoDuino

EcoDuino is a gardening kit from DFRobot. This kit consists of an MCU board and sensors. The board also provides an RF module breakout in an XBee breakout model. It enables the board to communicate with other platforms. The kit comes with soil moisture, temperature, and humidity (DHT11) sensors. If you're interested in this kit, you can buy it from this website: https://www.dfrobot.com/product-641.html.

The EcoDuino kit is shown in the following image:



Sensor devices for a smart gardening system

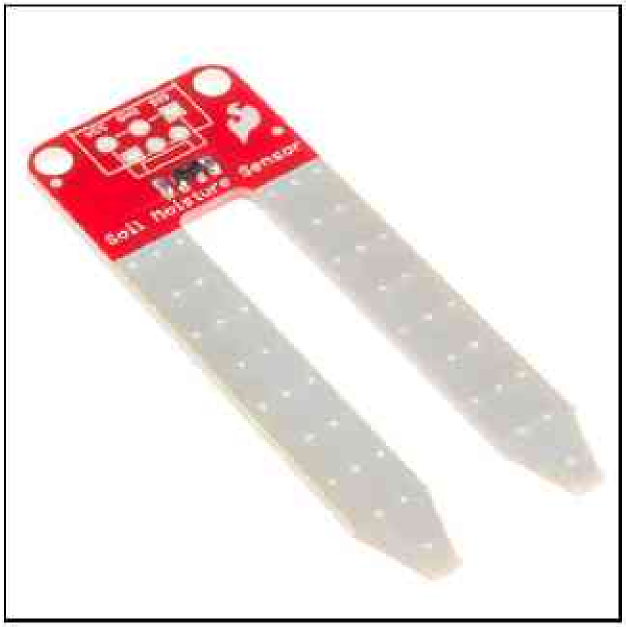
To build a smart gardening system, we need some sensors related to gardening and farming. In this section, we will review the main sensors that are used in gardening and farming. Soil moisture, temperature, and humidity are three parameters that we can use to build our smart gardening system.

Let's explore!

Soil moisture sensor

One of the parameters of gardening is soil moisture. We should measure soil moisture to ensure our plant grows well. There are many options when it comes to soil moisture sensors. You can use the SparkFun Soil Moisture Sensor, for example. You can find this module at https://www.sparkfun.com/products/13322 .

You can see the SparkFun Soil Moisture Sensor in the following image:



You can use other soil moisture sensors from cheap manufacturers in China. You can order them from Alibaba or Aliexpress.

To read soil moisture values, we can use analog I/O. To use analog I/O, you should be familiar with the following Sketch APIs:

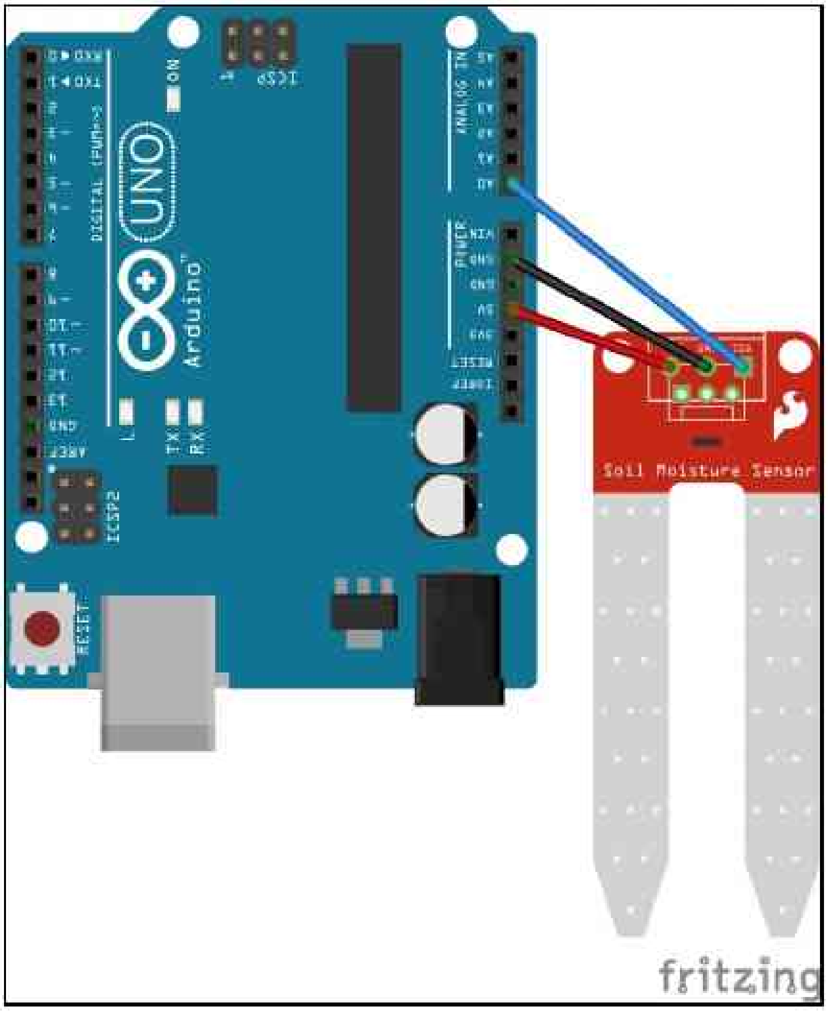
- `analogRead()` is for reading analog data from an analog pin on Arduino
- `analogWrite()` is for writing analog data to an analog pin on Arduino

For demonstration, we'll connect an Arduino board to the SparkFun Soil Moisture sensor. The following is the wiring used:

- VCC is connected to 5V on the Arduino

- GND is connected to GND on the Arduino
- SIG is connected to A0 on the Arduino

A complete wiring can be seen in the following figure:



Now you can open the Arduino software. If you haven't installed it yet, you can download and install the tool from https://www.arduino.cc. Once done, you can write a Sketch program. Create this script:

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int val;
  val = analogRead(A0);

  Serial.print("Soil moisture: ");
  Serial.print(val);
  delay(3000);
}
```

The program starts by initializing a `Serial` object with a baud rate of 9600. In a looping program, we read soil moisture levels using the `analogRead()` method. Then the measurement is printed to the serial port.

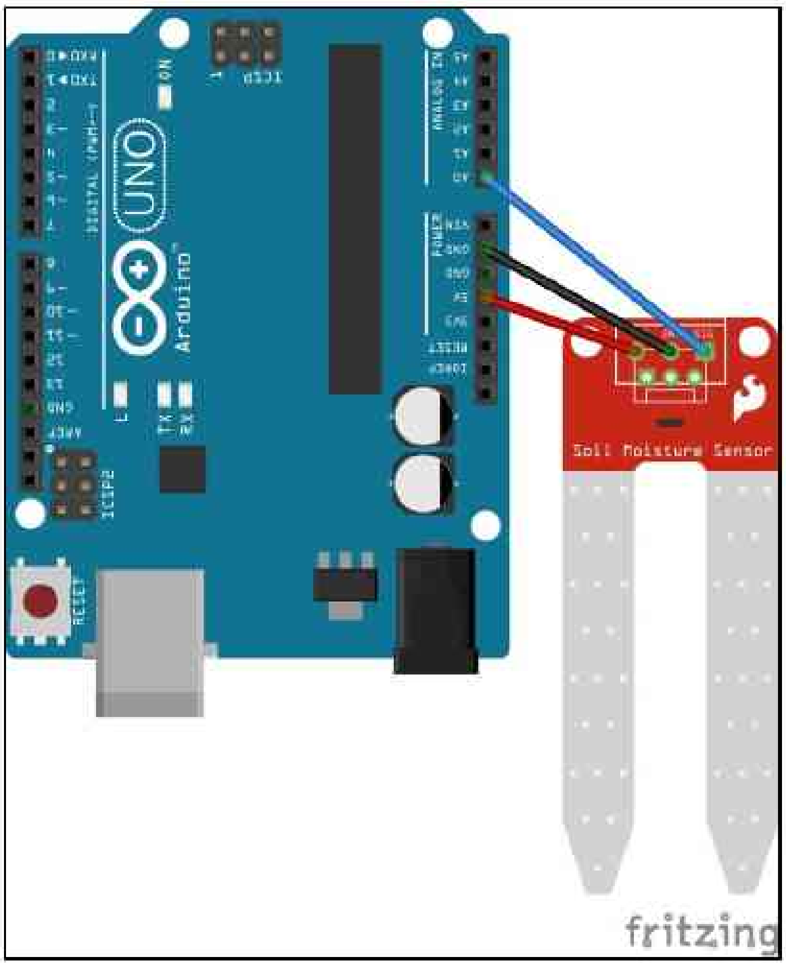
Save this Sketch file and upload the program to the Arduino board. To see the output data, you can use the serial monitor in the Arduino software. You can find it by going to **Tools | Serial Monitor**. You should see a soil moisture reading in the Serial Monitor tool.

Temperature and humidity sensor

Temperature and humidity have significant impact on the growth of the crop. Keeping temperature and humidity within certain values also keeps crops healthy. To monitor temperature and humidity, we can use the DHT22 or DHT11 for Arduino and Raspberry Pi. These sensors, DHT22 and DHT11, are famous sensors and have a lot of resources available for development.

The RHT03 (also known as DHT22) is a low-cost humidity and temperature sensor with a single-wire digital interface. You can obtain this module from SparkFun (https://www.sparkfun.com/products/10167) and Adafruit (https://www.adafruit.com/products/393). You may also find it in your local online or electronics store.

You can see the DHT22 module in the following figure:



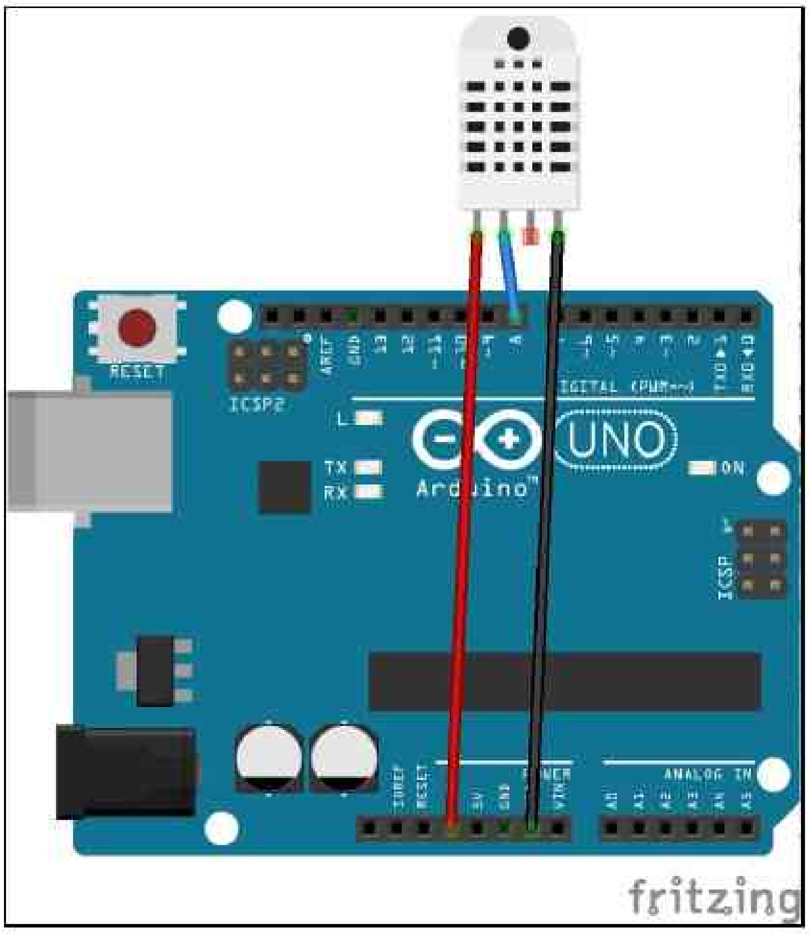
Source: https://www.adafruit.com/products/385

For further information about the DHT22 module, you can read the DHT22 datasheet at http://cdn.sparkfun.com/datasheets/Sensors/Weather/RHT03.pdf.

Now we connect the DHT22 module to the Arduino. The following is the wiring:

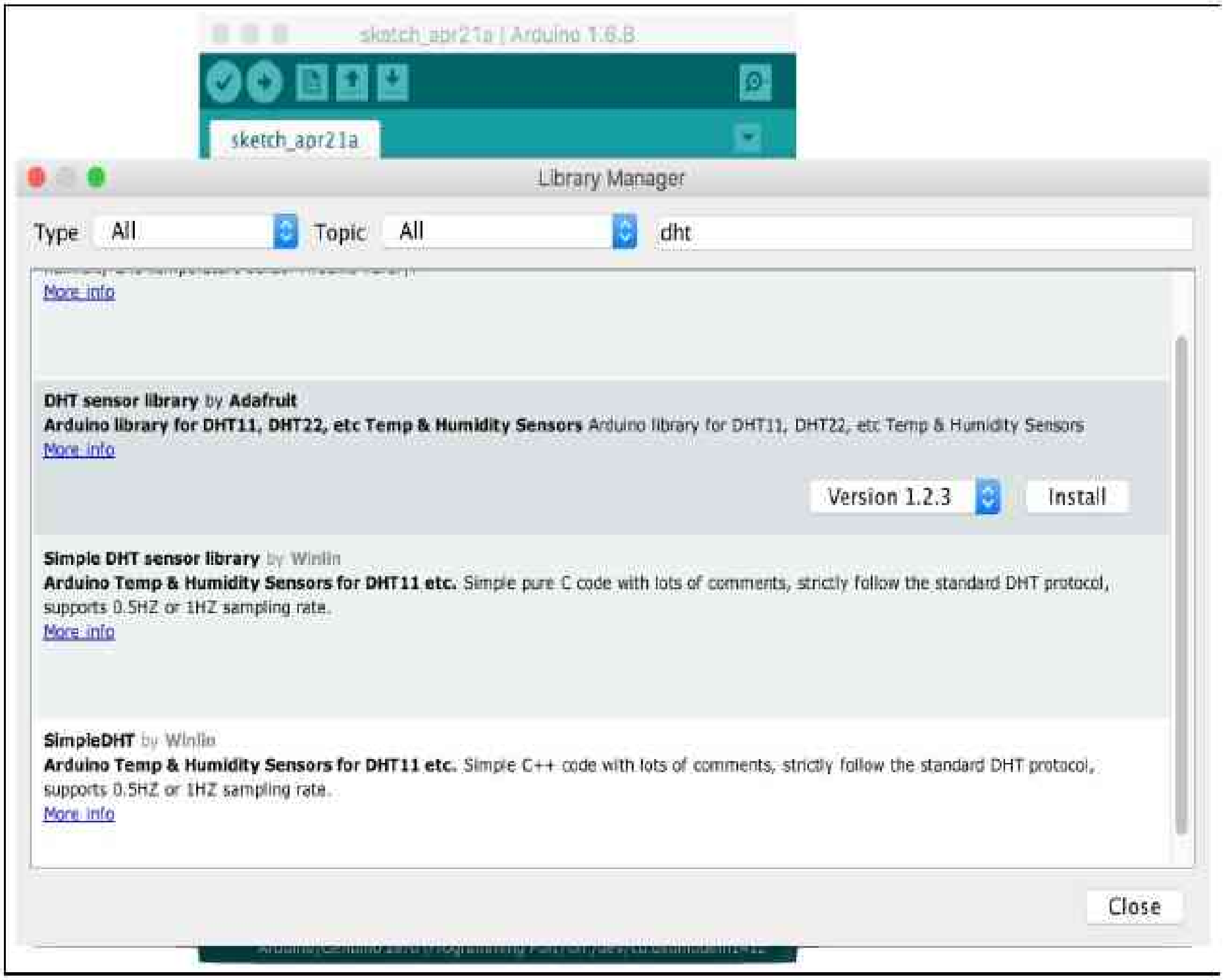
- VDD (pin 1) is connected to the 3.3V pin on the Arduino
- SIG (pin 2) is connected to digital pin 8 on the Arduino
- GND (pin 4) is connected to GND on the Arduino

You can see the complete wiring in the following figure:



To access the DHT-22 on the Arduino, we can use the DHT sensor library from Adafruit: https://github.com/adafruit/DHT-sensor-library. We can install this library from the Arduino software. Go to Sketch | Include Library | Manage Libraries and you will get a dialog.

Search for `dht` in Library Manager. You should see the DHT sensor library by Adafruit. Install this library:



DHT sensor library by **Adafruit**
Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors
More info

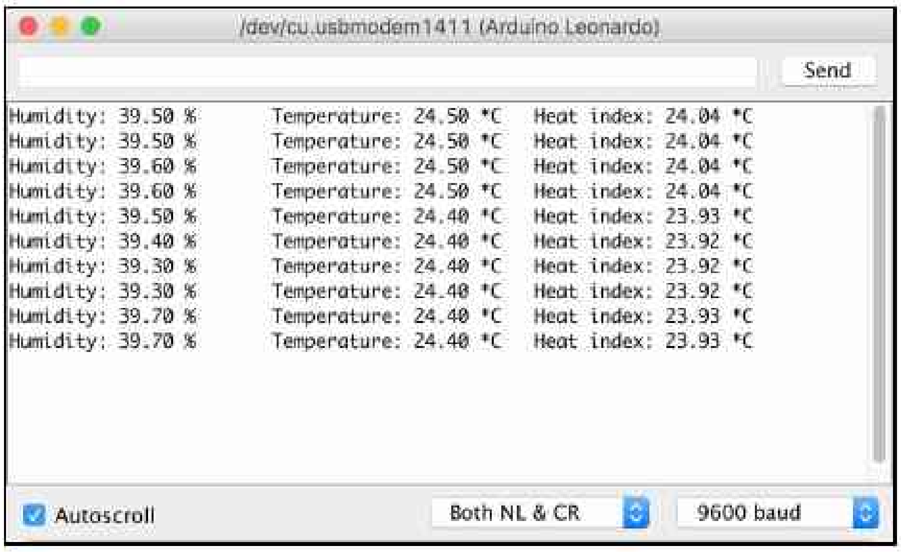
Simple DHT sensor library by **Winlin**
Arduino Temp & Humidity Sensors for DHT11 etc. Simple pure C code with lots of comments, strictly follow the standard DHT protocol, supports 0.5HZ or 1HZ sampling rate.
More info

SimpleDHT by **Winlin**
Arduino Temp & Humidity Sensors for DHT11 etc. Simple C++ code with lots of comments, strictly follow the standard DHT protocol, supports 0.5HZ or 1HZ sampling rate.
More info

Now let's start to write our Sketch program. You can develop a Sketch program to read temperature and humidity using `DHT22`. You can write this Sketch program in the Arduino software:

```
#include "DHT.h"

// define DHT22
#define DHTTYPE DHT22
// define pin on DHT22
#define DHTPIN 8

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
}

void loop() {
  delay(2000);

  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();


  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Compute heat index in Celsius (isFahreheit = false)
  float hic = dht.computeHeatIndex(t, h, false);

  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C\t");
  Serial.print("Heat index: ");
  Serial.print(hic);
  Serial.println(" *C ");
}
```

Save the program. Now you can compile and upload it to the Arduino board. Furthermore, you can open Serial Monitor to see sensor data from the serial port. You can see a sample program output in the following screenshot:



```
Humidity: 39.50 %	Temperature: 24.50 *C	Heat index: 24.04 *C
Humidity: 39.50 %	Temperature: 24.50 *C	Heat index: 24.04 *C
Humidity: 39.60 %	Temperature: 24.50 *C	Heat index: 24.04 *C
Humidity: 39.60 %	Temperature: 24.50 *C	Heat index: 24.04 *C
Humidity: 39.50 %	Temperature: 24.40 *C	Heat index: 23.93 *C
Humidity: 39.40 %	Temperature: 24.40 *C	Heat index: 23.92 *C
Humidity: 39.30 %	Temperature: 24.40 *C	Heat index: 23.92 *C
Humidity: 39.30 %	Temperature: 24.40 *C	Heat index: 23.92 *C
Humidity: 39.70 %	Temperature: 24.40 *C	Heat index: 23.93 *C
Humidity: 39.70 %	Temperature: 24.40 *C	Heat index: 23.93 *C
```

How it works

In the `setup()` function, we initialize the DHT module by calling `dht.begin()`. To read temperature and humidity, you can use `dht.readTemperature()` and `dht.readHumidity()`. You also can get a heat index using the `dht.computeHeatIndex()` function.

Watering your garden and farm

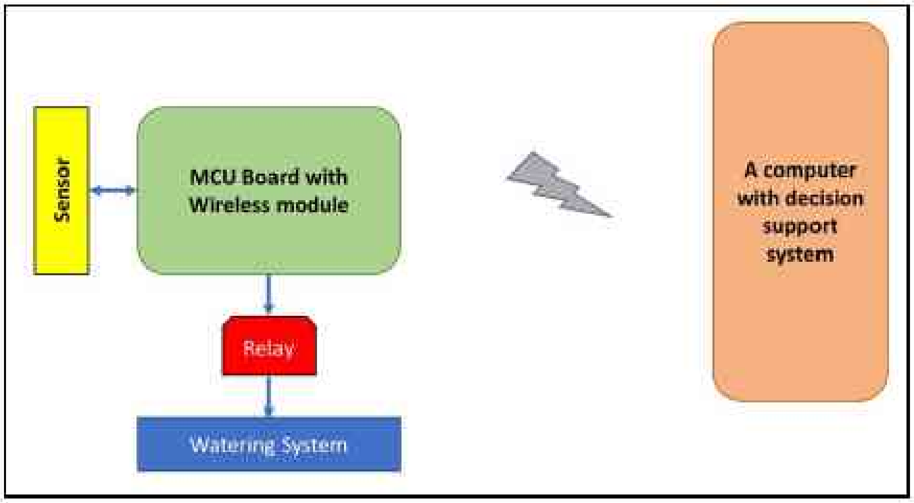
One of the problems in gardening systems is how to water our garden or farm. There are pumps that fit boards such as the Arduino and Raspberry Pi. The 350GPH liquid pump (https://www.sparkfun.com/products/10455) could be used on your Arduino or Raspberry Pi:

You can also use a higher-voltage pump. To control it from the Arduino, you can use a relay that is connected between the Arduino and the pump. A relay can be connected to digital pins from Arduino. Using `digitalRead()` and `digitalWrite()` we can communicate with Relay from Arduino.

Building a smart gardening system

In this section, we will develop a smart gardening system. We will use a PID controller to manage all inputs from our sensors, which will be used in decision system. We'll measure soil moisture, temperature, and humidity as parameters for our system. To keep it simple for now, we'll only use one parameter, soil moisture level.

A high-level architecture can be seen in the following figure:


Sensor
MCU Board with Wireless module
Relay
Watering System
A computer with decision support system

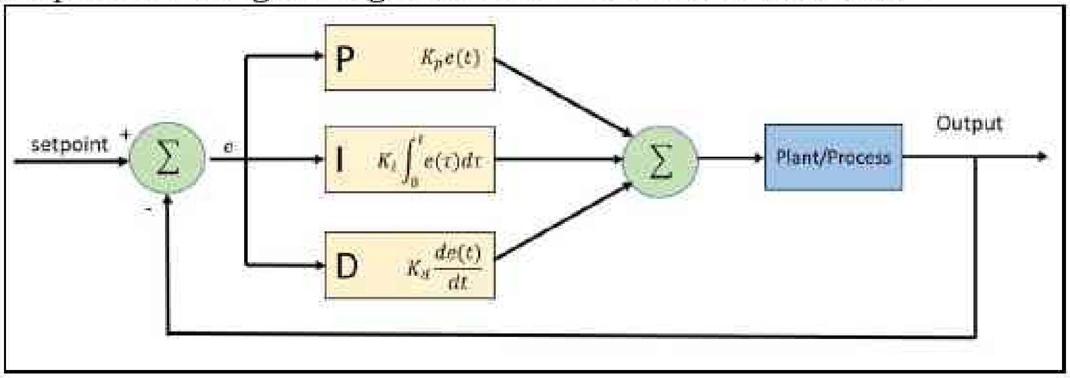
You can replace the MCU board and computer with a mini computer such as a Raspberry Pi. If you use a Raspberry Pi, you should remember that it does not have an analog input pin so you need an additional chip ADC, for instance the MCP3008, to work with analog input.

Assuming the watering system is connected via a relay, if we want to water the garden or farm, we just send digital value 1 to a relay. Some designs use a motor.

Let's build!

Introducing the PID controller

Proportional-integral-derivative (**PID**) control is the most common control algorithm used in the industry and has been universally accepted in industrial control. The basic idea behind a PID controller is to read a sensor and then compute the desired actuator output by calculating proportional, integral, and derivative responses and summing those three components to compute the output. The design of a general PID controller is as follows:



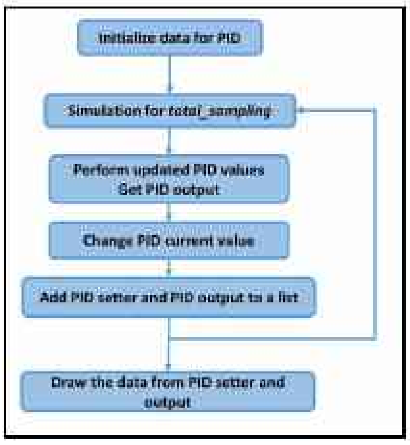
Furthermore, a PID controller formula can be defined as follows:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}$$

K_p, K_i, K_d represents the coefficients for the proportional, integral, and derivative. These parameters are non-negative values. The variable e represents the tracking error, the difference between the desired input value i and the actual output y. This error signal e will be sent to the PID controller.

Implementing a PID controller in Python

In this section, we'll build a Python application to implement a PID controller. In general, our program flowchart can be described as follows:


Initialize data for PID
Simulation for *total_sampling*
Perform updated PID values
Get PID output
Change PID current value
Add PID setter and PID output to a list
Draw the data from PID setter and output

We should not build a PID library from scratch. You can translate the PID controller formula into Python code easily. For implementation, I'm using the PID class from https://github.com/ivmech/ivPID. The following the `PID.py` file:

```
import time

class PID:
    """PID Controller
    """

    def __init__(self, P=0.2, I=0.0, D=0.0):

        self.Kp = P
        self.Ki = I
        self.Kd = D

        self.sample_time = 0.00
        self.current_time = time.time()
        self.last_time = self.current_time

        self.clear()

    def clear(self):
        """Clears PID computations and coefficients"""
        self.SetPoint = 0.0

        self.PTerm = 0.0
        self.ITerm = 0.0
        self.DTerm = 0.0
        self.last_error = 0.0

        # Windup Guard
        self.int_error = 0.0
        self.windup_guard = 20.0

        self.output = 0.0

    def update(self, feedback_value):
        """Calculates PID value for given reference feedback

        .. math::
            u(t) = K_p e(t) + K_i \int_{0}^{t} e(t)dt + K_d {de}/{dt}

        .. figure:: images/pid_1.png
           :align:   center

           Test PID with Kp=1.2, Ki=1, Kd=0.001 (test_pid.py)

        ""
        error = self.SetPoint - feedback_value

        self.current_time = time.time()
        delta_time = self.current_time - self.last_time
        delta_error = error - self.last_error

        if (delta_time >= self.sample_time):
            self.PTerm = self.Kp * error
            self.ITerm += error * delta_time

            if (self.ITerm < -self.windup_guard):
                self.ITerm = -self.windup_guard
            elif (self.ITerm > self.windup_guard):
                self.ITerm = self.windup_guard

            self.DTerm = 0.0
            if delta_time > 0:
                self.DTerm = delta_error / delta_time

            # Remember last time and last error for next calculation
            self.last_time = self.current_time
            self.last_error = error

            self.output = self.PTerm + (self.Ki * self.ITerm) + (self.Kd * self.DTerm)

    def setKp(self, proportional_gain):
        """Determines how aggressively the PID reacts to the current error with setting
        self.Kp = proportional_gain

    def setKi(self, integral_gain):
        """Determines how aggressively the PID reacts to the current error with setting
        self.Ki = integral_gain

    def setKd(self, derivative_gain):
        """Determines how aggressively the PID reacts to the current error with setting
        self.Kd = derivative_gain

    def setWindup(self, windup):
        """Integral windup, also known as integrator windup or reset windup,
       refers to the situation in a PID feedback controller where
        a large change in setpoint occurs (say a positive change)
and the integral terms accumulates a significant error         during the rise (windup
        to increase as this accumulated error is unwound
        (offset by errors in the other direction).
        The specific problem is the excess overshooting.
        """
        self.windup_guard = windup

    def setSampleTime(self, sample_time):
        """PID that should be updated at a regular interval.
        Based on a pre-determined sample time, the PID decides if it should compute or
        """
        self.sample_time = sample_time
```

For testing, we'll create a simple program for simulation. We need libraries such as `numpy`, `scipy`, `pandas`, `patsy`, and `matplotlib`. Firstly, you should `install python-dev` for Python development. Type these commands on a Raspberry Pi terminal:

```
$ sudo apt-get update
    $ sudo apt-get install python-dev
```

Now you can install the `numpy`, `scipy`, `pandas`, and `patsy` libraries. Open a Raspberry Pi terminal and type these commands:

```
$ sudo apt-get install python-scipy
    $ pip install numpy scipy pandas patsy
```

The last step is to install the `matplotlib` library from the source code. Type these commands into the Raspberry Pi terminal:

```
$ git clone https://github.com/matplotlib/matplotlib
    $ cd matplotlib
    $ python setup.py build
    $ sudo python setup.py install
```

After the required libraries are installed, we can test our `PID.py` code. Create a script with the following contents:

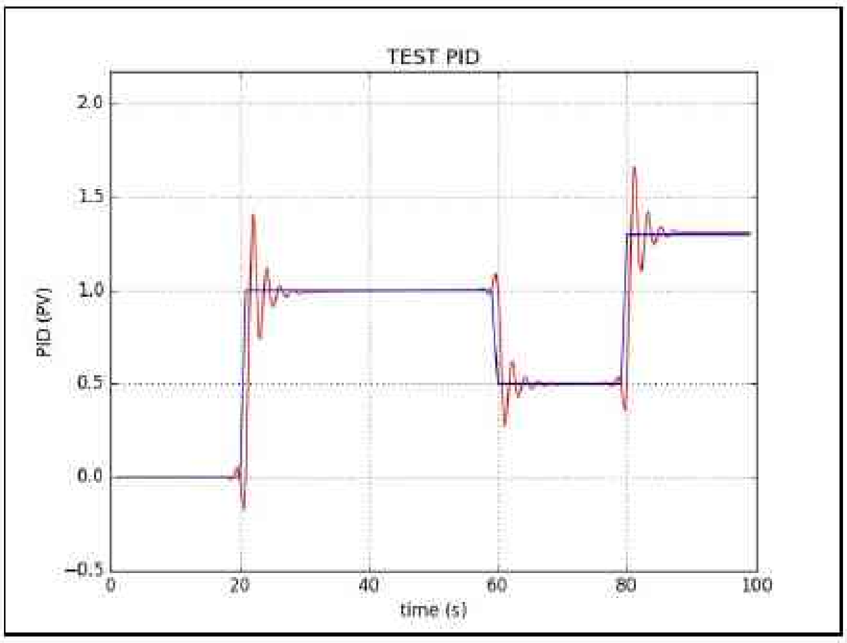
```
import matplotlib
matplotlib.use('Agg')

import PID
import time
import matplotlib.pyplot as plt
import numpy as np
from scipy.interpolate import spline


P = 1.4
I = 1
D = 0.001
pid = PID.PID(P, I, D)

pid.SetPoint = 0.0
pid.setSampleTime(0.01)

total_sampling = 100
```

```
feedback = 0

feedback_list = []
time_list = []
setpoint_list = []

print("simulating....")
for i in range(1, total_sampling):
    pid.update(feedback)
    output = pid.output
    if pid.SetPoint > 0:
        feedback += (output - (1 / i))

    if 20 < i < 60:
        pid.SetPoint = 1

    if 60 <= i < 80:
        pid.SetPoint = 0.5

    if i >= 80:
        pid.SetPoint = 1.3

    time.sleep(0.02)

    feedback_list.append(feedback)
    setpoint_list.append(pid.SetPoint)
    time_list.append(i)

time_sm = np.array(time_list)
time_smooth = np.linspace(time_sm.min(), time_sm.max(), 300)
feedback_smooth = spline(time_list, feedback_list, time_smooth)

fig1 = plt.gcf()
fig1.subplots_adjust(bottom=0.15)

plt.plot(time_smooth, feedback_smooth, color='red')
plt.plot(time_list, setpoint_list, color='blue')
plt.xlim((0, total_sampling))
plt.ylim((min(feedback_list) - 0.5, max(feedback_list) + 0.5))
plt.xlabel('time (s)')
plt.ylabel('PID (PV)')
plt.title('TEST PID')


plt.grid(True)
print("saving...")
fig1.savefig('result.png', dpi=100)
```

Save this program into a file called `test_pid.py`. Then run it:

```
$ python test_pid.py
```

This program will generate `result.png` as a result of the PID process. A sample output is shown in the following screenshot. You can see that the blue line has the desired values and the red line is the output of the PID:


TEST PID
PID (PV)
time (s)
2.0
1.5
1.0
0.5
0.0
−0.5
0
20
40
60
80
100

```
P = 1.4 <br/>I = 1 <br/>D = 0.001 <br/>pid = PID.PID(P, I, D) <br/>
<br/>pid.SetPoint = 0.0 <br/>pid.setSampleTime(0.01) <br/>
<br/>total_sampling = 100 <br/>feedback = 0 <br/>
<br/>feedback_list = [] <br/>time_list = [] <br/>setpoint_list = []

for i in range(1, total_sampling): <br/> pid.update(feedback) <br/>
output = pid.output <br/> if pid.SetPoint > 0: <br/> feedback +=
(output - (1 / i)) <br/> <br/> if 20 < i < 60: <br/> pid.SetPoint = 1
<br/> <br/> if 60 <= i < 80: <br/> pid.SetPoint = 0.5 <br/> <br/> if i
>= 80: <br/> pid.SetPoint = 1.3 <br/> <br/> time.sleep(0.02) <br/>
<br/> feedback_list.append(feedback) <br/>
setpoint_list.append(pid.SetPoint) <br/> time_list.append(i)

time_sm = np.array(time_list) <br/>time_smooth =
np.linspace(time_sm.min(), time_sm.max(), 300)
<br/>feedback_smooth = spline(time_list, feedback_list,
time_smooth) <br/> <br/>fig1 = plt.gcf()
<br/>fig1.subplots_adjust(bottom=0.15) <br/>
<br/>plt.plot(time_smooth, feedback_smooth, color='red')
<br/>plt.plot(time_list, setpoint_list, color='blue') <br/>plt.xlim((0,
total_sampling)) <br/>plt.ylim((min(feedback_list) - 0.5,
max(feedback_list) + 0.5)) <br/>plt.xlabel('time (s)')
<br/>plt.ylabel('PID (PV)') <br/>plt.title('TEST PID') <br/> <br/>
<br/>plt.grid(True) <br/>print("saving...")
<br/>fig1.savefig('result.png', dpi=100)
```

Sending data from the Arduino to the server

Not all Arduino boards have the capability to communicate with a server. Some Arduino models have built-in Wi-Fi that can connect and send data to a server, for instance, the Arduino Yun, Arduino MKR1000, and Arduino UNO Wi-Fi.

You can use the HTTP or MQTT protocols to communicate with the server. After the server receives the data, it will perform a computation to determine its decision.

Controlling soil moisture using a PID controller

Now we can change our PID controller simulation using a real application. We use soil moisture to decide whether to pump water. The output of the measurement is used as feedback input for the PID controller.

If the PID output is a positive value, then we turn on the watering system. Otherwise, we stop it. This may not be a good approach but is a good way to show how PID controllers work. Soil moisture data is obtained from the Arduino through a wireless network.

Let's write this program: import matplotlib

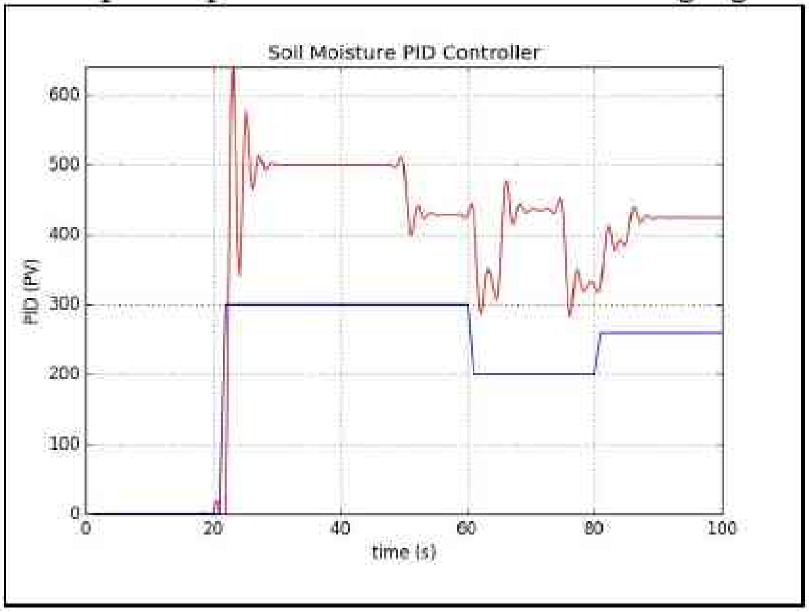
```
matplotlib.use('Agg')

import PID
import time
import matplotlib.pyplot as plt
import numpy as np
from scipy.interpolate import spline

P = 1.4
I = 1
D = 0.001
pid = PID.PID(P, I, D)

pid.SetPoint = 0.0
pid.setSampleTime(0.25) # a second

total_sampling = 100
sampling_i = 0
measurement = 0
feedback = 0
```

```
feedback_list = []
time_list = []
setpoint_list = []


def get_soil_moisture():
# reading from Arduino
# value 0 - 1023
return 200



print('PID controller is running..')
try:
while 1:
pid.update(feedback)
output = pid.output

soil_moisture = get_soil_moisture()
if soil_moisture is not None:

# # ## testing
# if 23 < sampling_i < 50:
# soil_moisture = 300

# if 65 <= sampling_i < 75:
# soil_moisture = 350

# if sampling_i >= 85:
# soil_moisture = 250
# # ################

if pid.SetPoint > 0:
feedback += soil_moisture + output

print('i={0} desired.soil_moisture={1:0.1f} soil_moisture={2:0.1f} pid.out=
{3:0.1f} feedback={4:0.1f}'
.format(sampling_i, pid.SetPoint, soil_moisture, output, feedback))
if output > 0:
print('turn on watering system')
elif output < 0:
print('turn off watering system')

if 20 < sampling_i < 60:
pid.SetPoint = 300 # soil_moisture

if 60 <= sampling_i < 80:
pid.SetPoint = 200 # soil_moisture

if sampling_i >= 80:
pid.SetPoint = 260 # soil_moisture



time.sleep(0.5)
sampling_i += 1

feedback_list.append(feedback)
setpoint_list.append(pid.SetPoint)
time_list.append(sampling_i)

if sampling_i >= total_sampling:
break

except KeyboardInterrupt:
print("exit")


print("pid controller done.")
print("generating a report...")
time_sm = np.array(time_list)
time_smooth = np.linspace(time_sm.min(), time_sm.max(), 300)
feedback_smooth = spline(time_list, feedback_list, time_smooth)
```

```
fig1 = plt.gcf()
fig1.subplots_adjust(bottom=0.15, left=0.1)

plt.plot(time_smooth, feedback_smooth, color='red')
plt.plot(time_list, setpoint_list, color='blue')
plt.xlim((0, total_sampling))
plt.ylim((min(feedback_list) - 0.5, max(feedback_list) + 0.5))
plt.xlabel('time (s)')
plt.ylabel('PID (PV)')
plt.title('Soil Moisture PID Controller')


plt.grid(True)
fig1.savefig('pid_soil_moisture.png', dpi=100)
print("finish")
```

Save this program to a file called `ch01_pid.py` and run it like this: **$ sudo python ch01_pid.py**

After executing the program, you should obtain a file called `pid_soil_moisture.png`. A sample output can be seen in the following figure:



How it works

Generally speaking, this program combines two things: reading the current soil moisture value through a soil moisture sensor and implementing a PID controller. After measuring the soil moisture level, we send the value to the PID controller program. The output of the PID controller will cause a certain action. In this case, it will turn on the watering machine.

Summary

We reviewed several gardening system platforms. Then we explored two sensor devices commonly used in real projects. Lastly, we built a decision system to automate for watering garden using PID.

In the next chapter, we will explore a smart parking system and try to build a prototype.

A Smart Parking System

Parking is one of the biggest problems in modern cities. A smart parking system is a solution to address vehicle parking in cities using the internet. This chapter will explore what a smart parking system is and how to build one using Arduino and Raspberry Pi boards.

In this chapter, we learn the following topics:

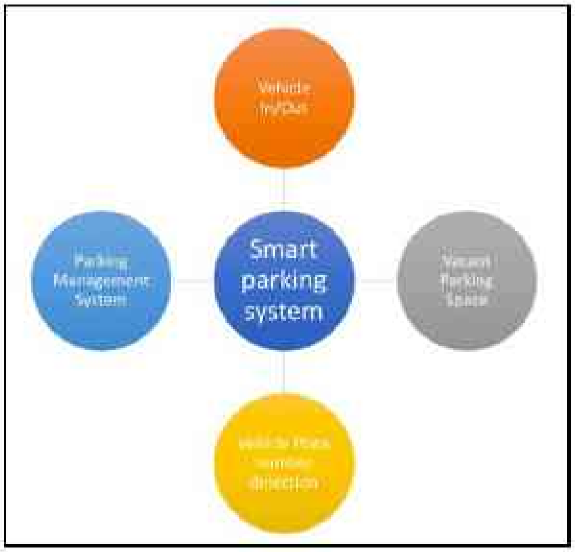
- Introducing smart parking systems
- Sensor devices for a smart parking system
- Vehicle entry/exit detection
- Vehicle plate number detection
- Vacant parking space detection
- Building a smart parking system

Let's get started!

Introducing smart parking systems

Parking has become one of the biggest problems in modern cities. When the vehicle growth rate is faster than street development rate, there will be issues with how these vehicles will be parked. A smart parking system becomes an alternative solution to addressing parking problems.

In general, a smart parking system will address problems shown in the following figure:



- **Vehicle In/Out** stands for detecting when a vehicle comes in and goes out. It will affect the number of vacant parking spaces and parking duration.
- **Vehicle plate number detection** aims at minimizing human errors in entering vehicle plate numbers into a system.
- **Vacant parking space** aims at optimizing parking space by knowing the number of vacant parking spaces.
- **Parking management system** is the main system to manage all activities in a parking system. It also provides information about vacant parking spaces to users through a website and mobile application.

In this chapter, we will explore several problems in parking systems and address them using automation.

Sensor devices for a smart parking system

To build a smart parking system, we should know about some sensors related to a smart parking system. In this section, we will review the main sensors that are used in one.

Let's explore!

Ultrasonic sensor - HC-SR04

The HC-SR04 is a cheap ultrasonic sensor. It is used to measure the range between itself and an object. Each HC-SR04 module includes an ultrasonic transmitter, a receiver, and a control circuit. You can see that the HC-SR04 has four pins: GND, VCC, Triger, and Echo. You can buy HC-SR04 from SparkFun, at https://www.sparkfun.com/products/13959, as shown in the next image. You can also find this sensor at SeeedStudio: https://www.seeedstudio.com/Ultra-Sonic-range-measurement-module-p-626.html. To save money, you can buy the HC-SR04 sensor from Aliexpress.

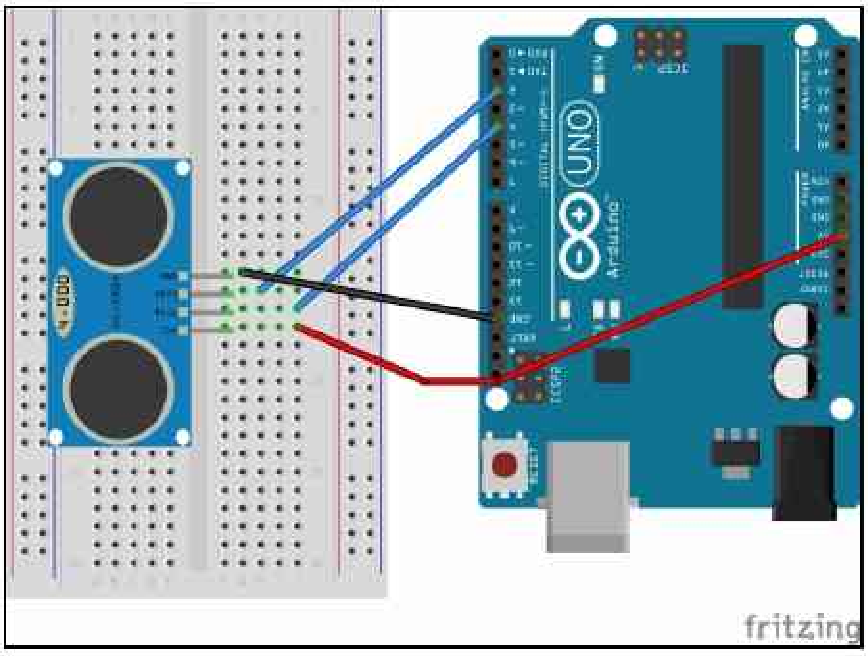
The HR-SR04 has four pins. There are VCC, GND, Echo, and Trigger. We can use it with Arduino, Raspberry Pi, or other IoT boards.

For demonstration, we can develop a simple application to access the HC-SR04 on an Arduino board. You can do the following wiring:

- HC-SR04 VCC is connected to Arduino 5V
- HC-SR04 GND is connected to Arduino GND
- HC-SR04 Echo is connected to Arduino Digital 2

- HC-SR04 Trigger is connected to Arduino Digital 4

You can see the hardware wiring in the following figure:



To work with the HC-SR04 module, we can use the NewPing library on our Sketch program. You can download it from http://playground.arduino.cc/Code/NewPing and then deploy it into the Arduino libraries folder. After it is deployed, you can start writing your Sketch program.

Now you can open your Arduino software. Then write the following complete code for our Sketch program:

```
#include <NewPing.h>

#define TRIGGER_PIN  2
#define ECHO_PIN     4
#define MAX_DISTANCE 600

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

long duration, distance;

void setup() {
  pinMode(13, OUTPUT);
  pinMode(TRIGGER_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  Serial.begin(9600);
}

void loop() {
  digitalWrite(TRIGGER_PIN, LOW);
  delayMicroseconds(2);

  digitalWrite(TRIGGER_PIN, HIGH);
  delayMicroseconds(10);

  digitalWrite(TRIGGER_PIN, LOW);
  duration = pulseIn(ECHO_PIN, HIGH);

  //Calculate the distance (in cm) based on the speed of sound.
  distance = duration/58.2;

  Serial.print("Distance=");
  Serial.println(distance);
  delay(200);

}
```

Save this sketch as `ArduinoHCSR04`.

This program will initialize a serial port at baudrate `9600` and activate digital pins for the HC-SR04 in the `setup()` function. In the `loop()` function, we read the value of the distance from the sensor and then display it on the serial port.

You can compile and upload this sketch to your Arduino board. Then you can open the Serial Monitor tool from the Arduino IDE menu. Make sure you set your baudrate to `9600`.

PIR motion sensor

A **Passive Infrared Detection** (**PIR**) motion sensor is used to object movement. We can use this sensor for detecting vehicle presence. PIR sensors can be found on SeeedStudio at https://www.seeedstudio.com/PIR-Motion-Sensor-Large-Lens-version-p-1976.html, Adafruit at https://www.adafruit.com/product/189, and SparkFun at https://www.sparkfun.com/products/13285. PIR motion sensors usually have three pins: VCC, GND, and OUT.

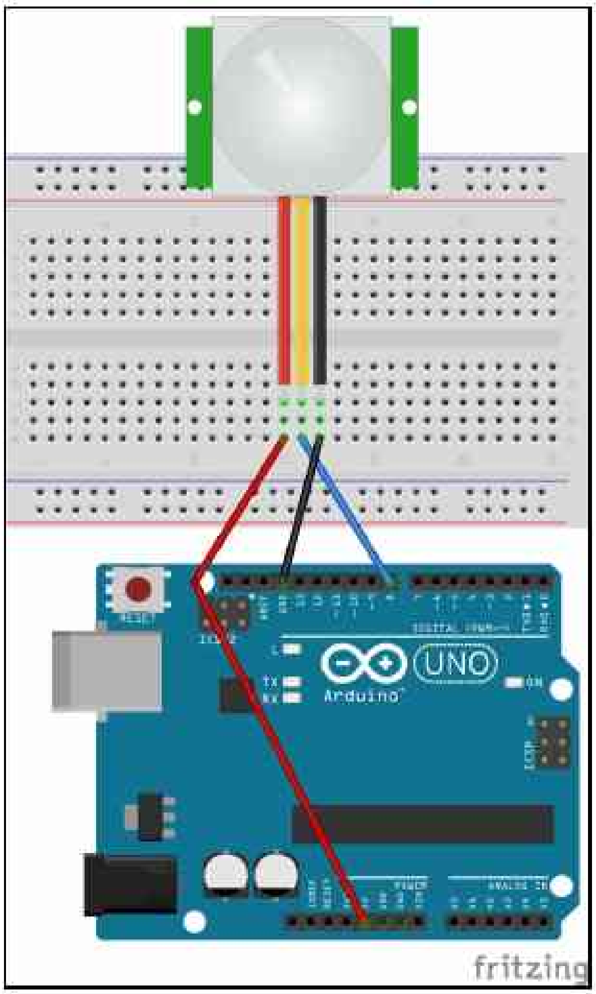
The OUT pin is digital output on which if we get value 1, it means motion is detected. You can see the PIR sensor from SeeedStudio here:

There's another PIR motion sensor from SparkFun, the SparkFun OpenPIR: https://www.sparkfun.com/products/13968. This sensor provides analog output so we can adjust the motion detection value. You can see the SparkFun OpenPIR here:

For testing, we develop an Arduino application to access the PIR motion sensor. We will detect object motion using the PIR motion sensor. You can perform the following hardware wiring:

- PIR Sensor VCC is connected to Arduino 5V
- PIR Sensor GND is connected to Arduino GND
- PIR Sensor OUT is connected to Arduino Digital 8

You can see the complete wiring here:



Now we can write a sketch program using Arduino software. To read object motion detection in digital form, we can use `digitalRead()`. If the program detects object motion, we turn on the LED. Otherwise, we turn it off.

Here is the complete sketch:

```
#define PIR_LED 13 // LED
#define PIR_DOUT 8 // PIR digital output on D8

int val = 0;
int state = LOW;
void setup()
{
  pinMode(PIR_LED, INPUT);
  pinMode(PIR_DOUT, INPUT);
  Serial.begin(9600);
}

void loop()
{
  val = digitalRead(PIR_DOUT);
  if(val==HIGH)
  {
    digitalWrite(PIR_LED, HIGH);
    if(state==LOW) {
      Serial.println("Motion detected!");
      state = HIGH;
    }
  }else{
    digitalWrite(PIR_LED, LOW);
    if (state == HIGH){
      Serial.println("Motion ended!");
      state = LOW;
    }
  }
}
```

Save this sketch as `ArduinoPIR`.

You can compile and upload the program to your Arduino board. To see the program output, you can open the Serial Monitor tool from Arduino. Now you test motion detection using your hand and see the program output in the Serial Monitor tool.

Next, we will use the SparkFun OpenPIR or other PIR motion sensor. Based on its datasheet, the SparkFun OpenPIR has similar pins similar to PIR motion sensors. However, the SparkFun OpenPIR sensor has an additional pin, the AOUT analog output.

This pin provides analog output as a voltage value from the object's motion level. From this case, we set the object motion-detection level based on the analog output.

For testing, we'll develop a sketch program using the SparkFun OpenPIR and Arduino. You can build the following hardware wiring:

- OpenPIR Sensor VCC is connected to Arduino 5V
- OpenPIR Sensor GND is connected to Arduino GND
- OpenPIR Sensor AOUT is connected to Arduino Analog A0
- OpenPIR Sensor OUT is connected to Arduino Digital 2

Now we'll modify the code sample from SparkFun. To read analog input, we can use `analogRead()` in our sketch program. Write this sketch for our demo:

```
#define PIR_AOUT A0  // PIR analog output on A0
#define PIR_DOUT 2   // PIR digital output on D2
#define LED_PIN  13  // LED to illuminate on motion
```

```
void setup()
{
  Serial.begin(9600);
  // Analog and digital pins should both be set as inputs:
  pinMode(PIR_AOUT, INPUT);
  pinMode(PIR_DOUT, INPUT);

  // Configure the motion indicator LED pin as an output
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);
}

void loop()
{
  int motionStatus = digitalRead(PIR_DOUT);
  if (motionStatus == HIGH)
    digitalWrite(LED_PIN, HIGH); // motion is detected
  else
    digitalWrite(LED_PIN, LOW);

  // Convert 10-bit analog value to a voltage
  unsigned int analogPIR = analogRead(PIR_AOUT);
  float voltage = (float) analogPIR / 1024.0 * 5.0;
  Serial.print("Val: ");
  Serial.println(voltage);
}
```

Save it as `ArduinoOpenPIR`.

You can compile and upload the sketch to your Arduino board. Open the Serial Monitor tool to see the program output. Now you can move your hand and see the program output in the Serial Monitor tool.

Hall Effect sensor

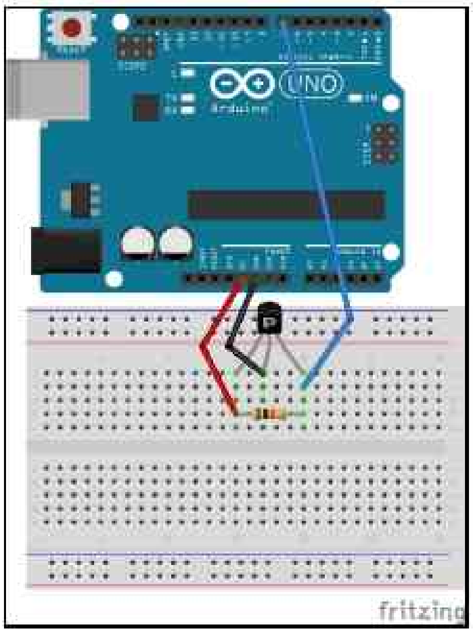
The Hall Effect sensor works on the principle of the Hall Effect of magnetic fields. The sensor will generate a voltage that indicates whether the sensor is in the proximity of a magnet or not. This sensor is useful to detect vehicular presence, especially vehicles coming in and going out of the parking area.

There are ICs and modules that implement the Hall Effect. For instance, you can use a transistor, US1881. You can read about and buy this sensor from SparkFun (https://www.sparkfun.com/products/9312). This transistor may also be available in your local electronic store:

Now we can try to develop sketch program to read a magnetic value from a Hall Effect sensor. We will use the US1881. You can also use any Hall Effect chip according to your use case. Firstly, we'll make our hardware wiring as follows:

- US1881 VCC is connected to Arduino 5V
- US1881 GND is connected to Arduino GND
- US1881 OUT is connected to Arduino Digital 7
- A resistor of 10K ohms is connected between US1881 VCC and US1881 OUT

You can see the complete hardware wiring here:



We can now start writing our sketch program. We'll use `digitalRead()` to indicate a magnet's presence. If the digital value is `LOW`, we'll detect the magnet's presence. Write this program:

```
int Led = 13 ;
int SENSOR = 7 ;
int val;

void setup ()
{
  Serial.begin(9600);
  pinMode(Led, OUTPUT) ;
  pinMode(SENSOR, INPUT) ;
}

void loop ()
{
  val = digitalRead(SENSOR);
  if (val == LOW)
  {
    digitalWrite (Led, HIGH);
    Serial.println("Detected");
  }
  else
  {
    digitalWrite (Led, LOW);
  }
  delay(1000);
}
```

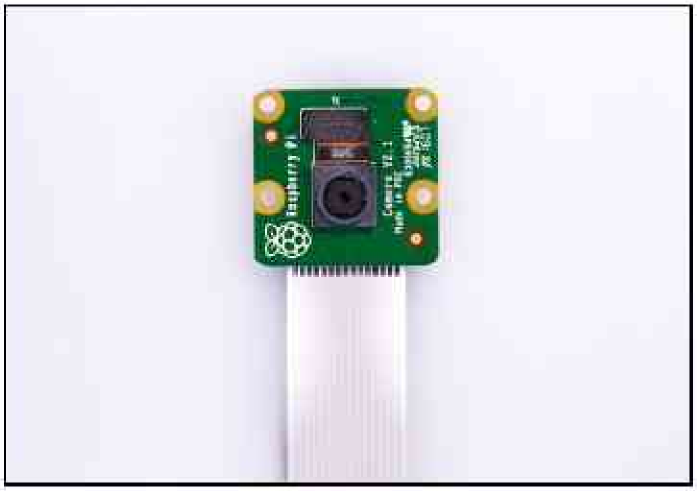
Save this sketch as `ArduinoHallEffect`.

You can compile and upload the program to your Arduino board. To see the program output, you can open the Serial Monitor tool from Arduino. For testing, you can use a magnet and put it next to the sensor. You should see "Detected" on the Serial Monitor tool.

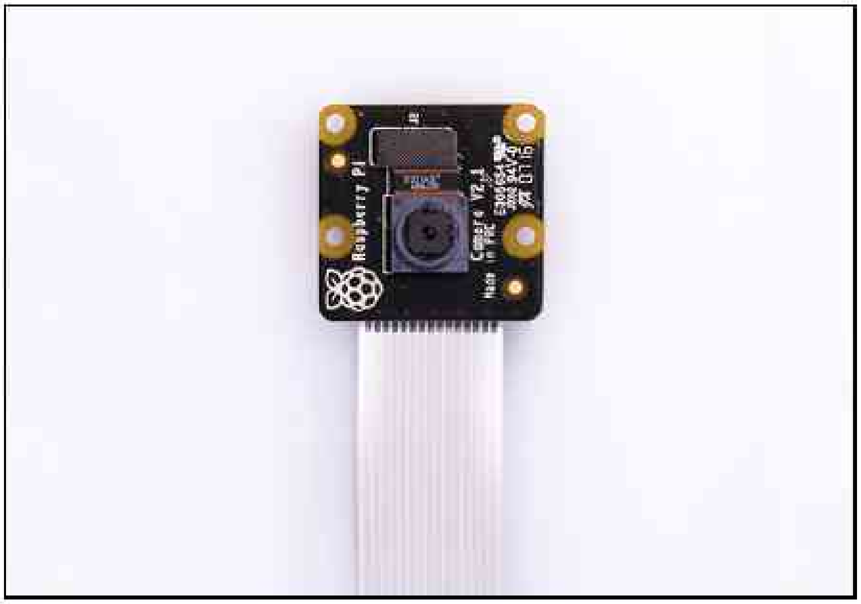
Camera

A camera is a new approach for parking sensors. Using image processing and machine learning, we can detect vehicles coming in to/going out of a parking area. A camera is also useful for a vehicle's plate number detection.

If we implement a camera with a Raspberry Pi, we have many options. The Raspberry Pi Foundation makes the official camera for Raspberry Pi. Currently, the recommended camera is the camera module version 2. You can read about and buy it from https://www.raspberrypi.org/products/camera-module-v2/. You can see the the Pi Camera module V2 here:



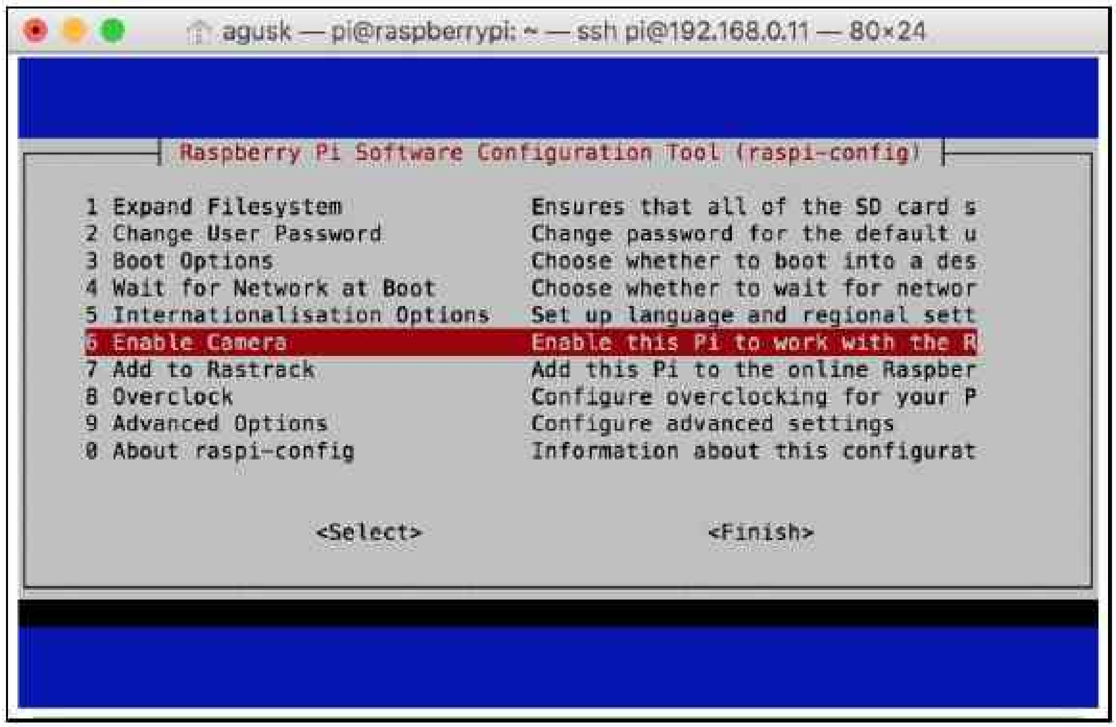
If you are working in low light conditions, the Pi Camera module V2 is not good at taking photos and recording. The Raspberry Pi Foundation provides the Pi Camera Noire V2. This camera can work in low light. You can find this product at https://www.raspberrypi.org/products/pi-noir-camera-v2/. You can see the Pi Camera Noire V2 here:



This camera module is connected to the Raspberry Pi through the CSI interface. To use this camera, you should activate it in your OS. For instance, if we use the Raspbian OS, we can activate the camera from the command line. You can type this command:

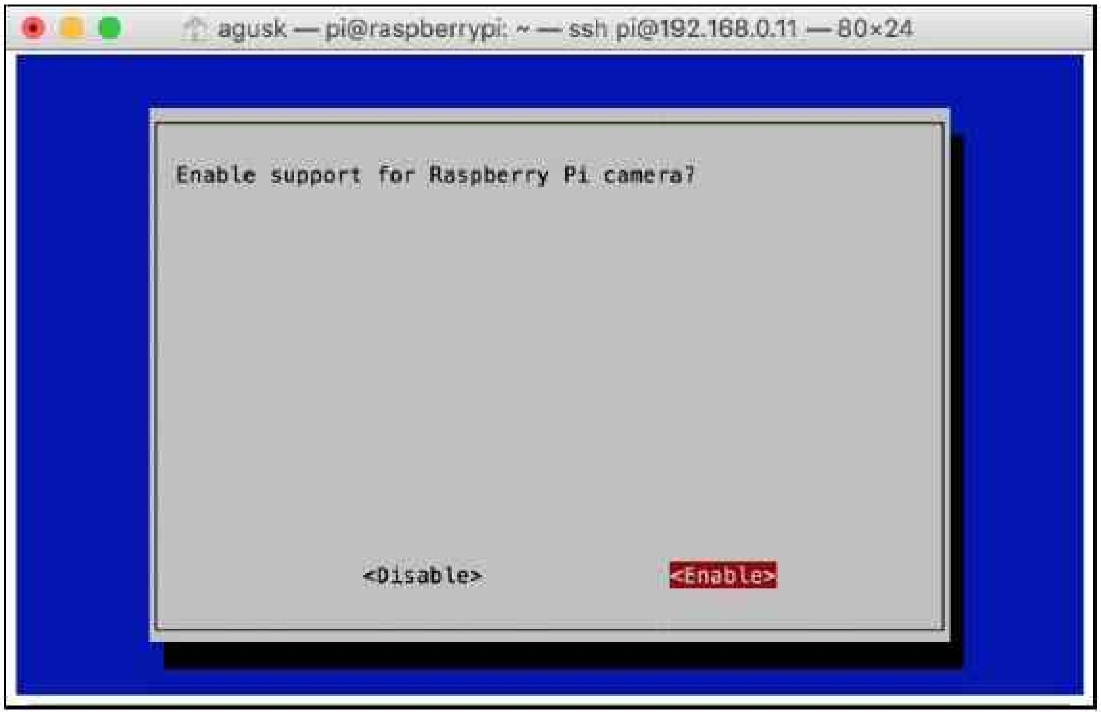
```
$ sudo apt-get update
    $ sudo raspi-config
```

After it is executed, you should get the following menu:



```
agusk — pi@raspberrypi: ~ — ssh pi@192.168.0.11 — 80×24

Raspberry Pi Software Configuration Tool (raspi-config)

 1 Expand Filesystem              Ensures that all of the SD card s
 2 Change User Password           Change password for the default u
 3 Boot Options                   Choose whether to boot into a des
 4 Wait for Network at Boot       Choose whether to wait for networ
 5 Internationalisation Options   Set up language and regional sett
 6 Enable Camera                  Enable this Pi to work with the R
 7 Add to Rastrack                Add this Pi to the online Raspber
 8 Overclock                      Configure overclocking for your P
 9 Advanced Options               Configure advanced settings
 0 About raspi-config             Information about this configurat

              <Select>                      <Finish>
```

Select `6 Enable Camera` to activate the camera on the Raspberry Pi. You'll get a confirmation to enable the camera module, as shown here:



```
agusk — pi@raspberrypi: ~ — ssh pi@192.168.0.11 — 80×24

Enable support for Raspberry Pi camera?

        <Disable>            <Enable>
```

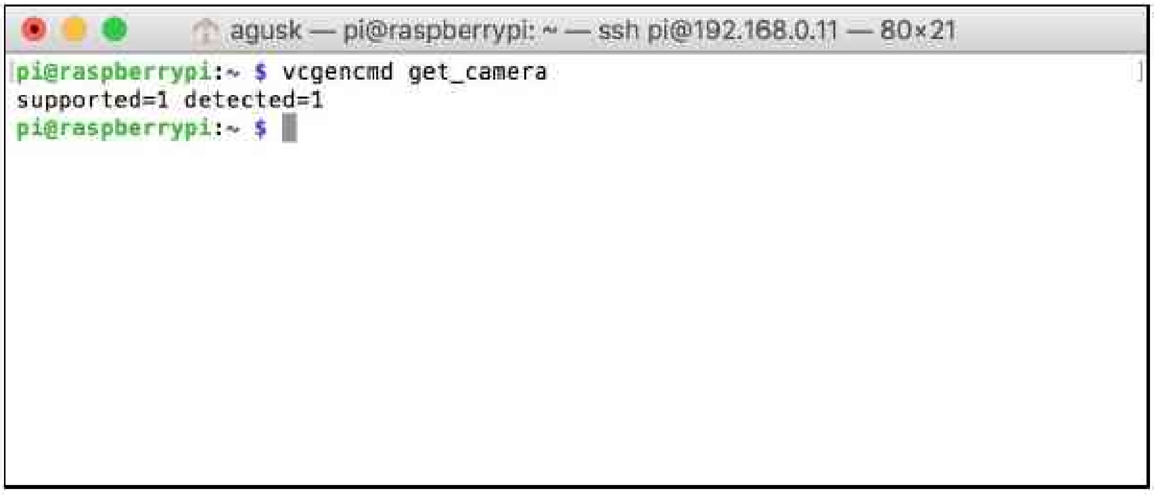
After this, Raspbian will reboot.

Now you can attach the Pi camera on the CSI interface. You can see my Pi camera here:

After it's attached, it's recommended you reboot your Raspberry Pi. If rebooting is done, you can verify whether the camera on the CSI interface is detected or not. You can use `vcgencmd` to check it. Type this command:

```
$ vcgencmd get_camera
```

You should see your camera detected in the terminal. For instance, you can see my response from the Pi camera here:



```
pi@raspberrypi:~ $ vcgencmd get_camera
supported=1 detected=1
pi@raspberrypi:~ $
```

For testing, we'll develop a Python application to take a photo and save it to local storage. We'll use the `PiCamera` object to access the camera.

Use this script:

```
from picamera import PiCamera
from time import sleep

camera = PiCamera()

camera.start_preview()
sleep(5)
camera.capture('/home/pi/Documents/image.jpg')
camera.stop_preview()
print("capture is done")
```

Save it into a file called `picamera_still.py`. Now you can run this file on your Terminal by typing this command:

```
$ python picamera_still.py
```

If you succeed, you should see an image file, `image.jpg`:

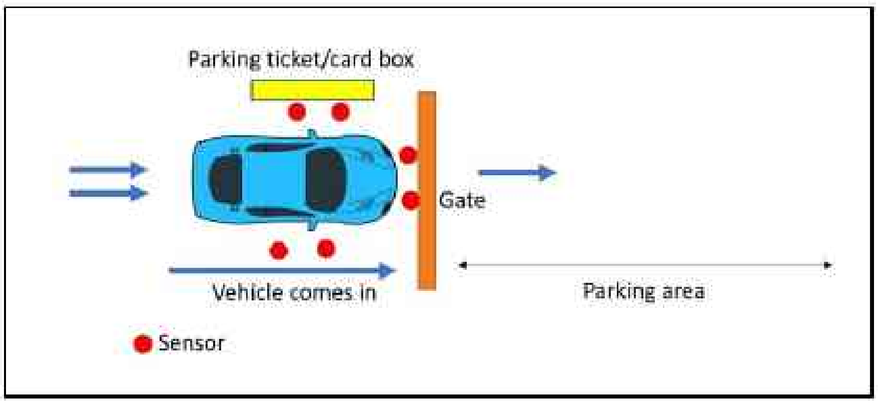
Vehicle entry/exit detection

A parking lot usually has an in-gate and out-gate. When a vehicle comes in, a driver presses a button to a get parking ticket. Then, the gate opens. Some parking places use a card as driver identity. After it is tapped, the gate will open.

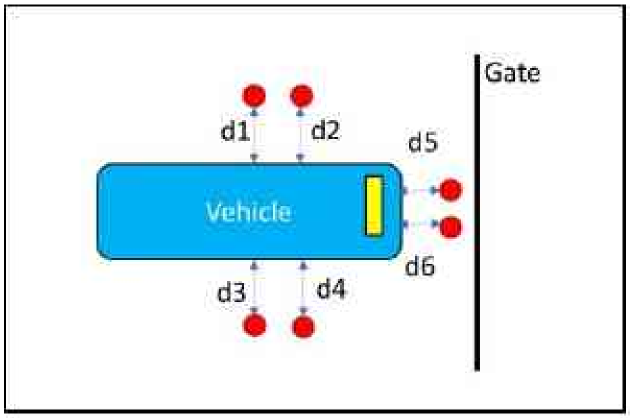
Here's a real parking gate:

Source: http://queenland.co.id/secure-parking/

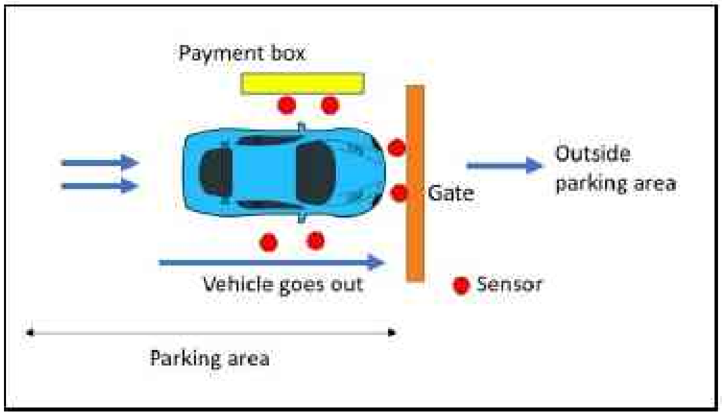
To build a parking in-gate, we can design one as shown in the following figure. We put sensor devices on the left, right, and a front of a vehicle. You can use sensor devices that we have learned about.



For instance, we can use an ultrasonic sensor. We'll put some sensor devices on the right, left, and front of the vehicle. The sensor device position in the gate is shown in the following figure. All sensors acquire and obtain distance values, dx. Then, we set a threshold value from a collection of distance values to indicate whether a vehicle is present.



The out-gate has a similar design like the parking in-gate. You should modify the gate and sensor position based on the gate's position and size. Each gate will be identified as an in-gate or out-gate.



A gate will open if the driver has taken a parking ticket or taps a parking card. This scenario can be made automatic. For instance, we put a camera in front of the vehicle. This camera will detect the vehicle plate number. After detecting the plate number of the vehicle, the parking gate will open.

We will learn how to detect and identify plate numbers on vehicles using a camera in the next section.

Vehicle plate number detection

A smart parking system ensures all activities take place automatically. Rather than reading a vehicle plate number manually, we can make it read a vehicle plate number using a program through a camera. We can use the OpenALPR library for reading vehicle plate numbers.

OpenALPR is an open source automatic license plate recognition library written in C++ with bindings in C#, Java, Node.js, Go, and Python. You can visit the official website at https://github.com/openalpr/openalpr.

We will install OpenALPR and test it with vehicle plate numbers in the next section.

Installing OpenALPR

In this section, we will deploy OpenALPR on the Raspberry Pi. Since OpenALPR needs more space on disk, make sure your Raspberry has space. You may need to expand your storage size.

Before we install OpenALPR, we need to install some prerequisite libraries to ensure correct installation. Make sure your Raspberry Pi is connected to the internet. You can type these commands to install all prerequisite libraries:

```
$ sudo apt-get update
    $ sudo apt-get upgrade
    $ sudo apt-get install autoconf automake libtool
    $ sudo apt-get install libleptonica-dev
    $ sudo apt-get install libicu-dev libpango1.0-dev libcairo2-dev
    $ sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavforma
    $ sudo apt-get install python-dev python-numpy libjpeg-dev libpng-dev libtiff-dev l
    $ sudo apt-get install libgphoto2-dev
    $ sudo apt-get install virtualenvwrapper
    $ sudo apt-get install liblog4cplus-dev
    $ sudo apt-get install libcurl4-openssl-dev
    $ sudo apt-get install autoconf-archive
```

Once this is done, we should install the required libraries, such as Leptonica, Tesseract, and OpenCV. We will install them in the next few steps.

In this step, we'll install the Leptonica library. It's a library to perform image processing and analysis. You can get more information about the Leptonica library from the official website at http://www.leptonica.org. To install the library on the Raspberry Pi, we'll download the source code from Leptonica and then compile it. Right now, we'll install the latest version of the Leptonica library, version 1.74.1. You can use these commands:

```
$ wget http://www.leptonica.org/source/leptonica-1.74.1.tar.gz
    $ gunzip leptonica-1.74.1.tar.gz
    $ tar -xvf leptonica-1.74.1.tar
    $ cd leptonica-1.74.1/
    $ ./configure
    $ make
    $ sudo make install
```

This installation takes more time, so ensure all prerequisite libraries have been installed properly.

After installing the Leptonica library, we should install the Tesseract library. This library is useful for processing OCR images. We'll install the Tesseract library from its source code on https://github.com/tesseract-ocr/tesseract. The following is a list of commands to install the Tesseract library from source:

```
$ git clone https://github.com/tesseract-ocr/tesseract
    $ cd tesseract/
    $ ./autogen.sh
    $ ./configure
    $ make -j2
    $ sudo make install
```

This installation process takes time. After it is done, you also can install data training files for the Tesseract library using these commands:

```
$ make training
    $ sudo make training-install
```

The last required library to install OpenALPR is OpenCV. It's a famous library for computer vision. In this scenario, we will install OpenCV from source. It takes up more space on disk, so your Raspberry Pi disk should have enough free space. For demonstration, we'll install the latest OpenCV library. Version 3.2.0 is available on https://github.com/opencv/opencv. You can find the contribution modules on https://github.com/opencv/opencv_contrib. We also need to install modules with the same version as our OpenCV version.

First, we'll download and extract the OpenCV library and its modules. Type these commands:

```
$ wget https://github.com/opencv/opencv/archive/3.2.0.zip
    $ mv 3.2.0.zip opencv.zip
    $ unzip opencv.zip
    $ wget https://github.com/opencv/opencv_contrib/archive/3.2.0.zip
    $ mv 3.2.0.zip opencv_contrib.zip
    $ unzip opencv_contrib.zip
```

Now we'll build and install OpenCV from source. Navigate your Terminal to the folder where the OpenCV library was extracted. Let's suppose our OpenCV contribution module has been installed at `~/Downloads/opencv_contrib-3.2.0/modules`. You can change this based on where your OpenCV contribution module folder was extracted.

```
$ cd opencv-3.2.0/
    $ mkdir build
    $ cd build/
    $ cmake -D CMAKE_BUILD_TYPE=RELEASE \
        -D CMAKE_INSTALL_PREFIX=/usr/local \
        -D INSTALL_PYTHON_EXAMPLES=ON \
        -D OPENCV_EXTRA_MODULES_PATH=~/Downloads/opencv_contrib-3.2.0/modules \
        -D BUILD_EXAMPLES=ON ..
    $ make
    $ sudo make install
    $ sudo ldconfig
```

This installation takes a while. It may take over two hours. Make sure there is no error in the installation process.

Finally, we'll install OpenALPR from source:

```
$ git clone https://github.com/openalpr/openalpr.git
    $ cd openalpr/src/
    $ cmake ./
    $ make
    $ sudo make install
    $ sudo ldconfig
```

After all this is installed, you are ready for testing. If it's not, resolve your errors before moving to the testing section.

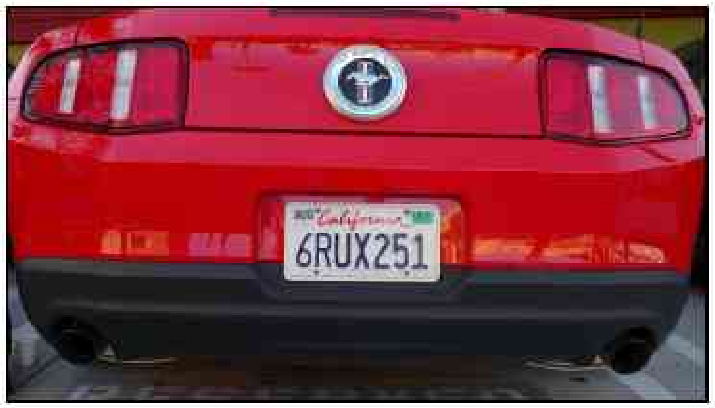
Testing vehicle plate number detection

Now that we've installed OpenALPR on our Raspberry Pi, we can test it using several vehicle plate numbers.

For instance, we have a plate number image here. The number is `6RUX251`. We can call `alpr`:

```
$ alpr lp.jpg
    plate0: 10 results
        - 6RUX251      confidence: 95.1969
        - 6RUXZ51      confidence: 87.8443
        - 6RUX2S1      confidence: 85.4606
        - 6RUX25I      confidence: 85.4057
        - 6ROX251      confidence: 85.3351
        - 6RDX251      confidence: 84.7451
        - 6R0X251      confidence: 84.7044
        - 6RQX251      confidence: 82.9933
        - 6RUXZS1      confidence: 78.1079
        - 6RUXZ5I      confidence: 78.053
```

You can see the that the result is **6RUX251** with a confidence of 95.1969%.



Source image: http://i.imgur.com/pjukRD0.jpg

The second demo for testing is a plate number from Germany, in the file `Germany_BMW.jpg`. You can see it in the next image. The number is `BXG505`. Let's test it using `aplr`:

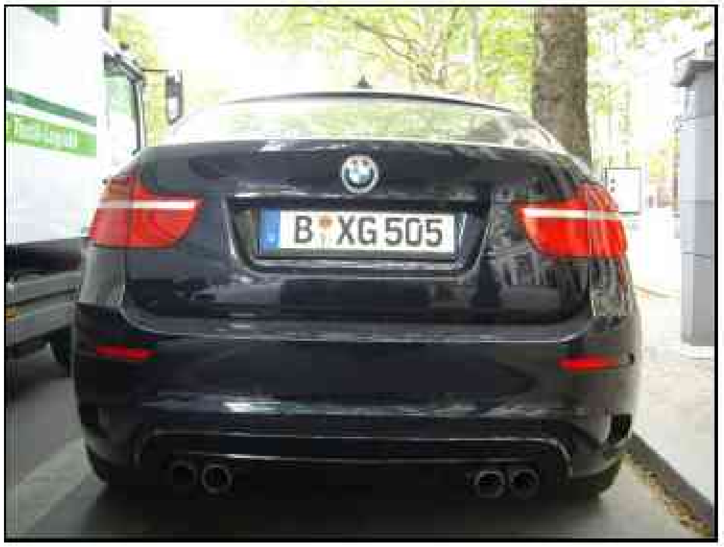
```
$ alpr Germany_BMW.jpg
```

```
    plate0: 10 results
        - 3XG5C  confidence: 90.076
        - XG5C   confidence: 82.3237
        - SXG5C  confidence: 80.8762
        - BXG5C  confidence: 79.8428
        - 3XG5E  confidence: 78.3885
        - 3XGSC  confidence: 77.9704
        - 3X65C  confidence: 76.9785
        - 3XG5G  confidence: 72.5481
        - 3XG5   confidence: 71.3987
        - XG5E   confidence: 70.6362
```

You can see that the result is not good. Now we'll specify that it is a plate number from Europe by passing the `eu` parameter.

```
$ alpr -c eu Germany_BMW.jpg
    plate0: 10 results
        - B0XG505      confidence: 91.6681
        - BXG505             confidence: 90.5201
        - BOXG505      confidence: 89.6625
        - B0XG5O5      confidence: 84.2605
        - B0XG5D5      confidence: 83.6188
        - B0XG5Q5      confidence: 83.1674
        - BXG5O5             confidence: 83.1125
        - B0XG5G5      confidence: 82.485
        - BXG5D5             confidence: 82.4708
        - BOXG5O5      confidence: 82.2548
```

Now you can see that the plate number is detected correctly with a confidence 90.5201%.



Source image: http://vehicleplatex6.blogspot.co.id/2012/06/

Vacant parking space detection

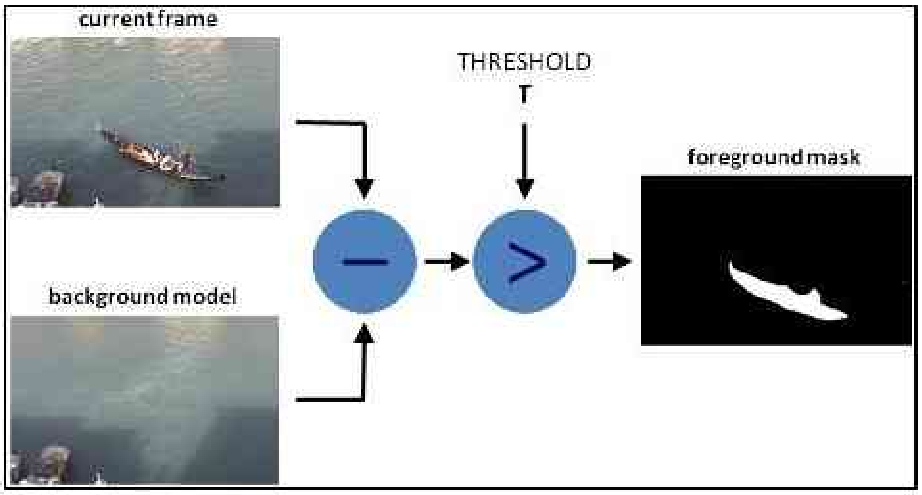
In a smart parking system, information about vacant parking spaces is important. A simple method to identify how many vacant spaces are available is to count the difference between the parking lot capacity and the number of vehicles currently in the lot. This approach has a disadvantage because we don't know which parking areas are vacant. Suppose a parking lot has four levels. We should know whether a parking area level is full or not.

Suppose we have a parking area as shown in the next image. We can identify how many vacant parking spaces there are using image processing and pattern recognition.

Source image: https://www.hotels.com/ho557827/jakarta-madrix-jakarta-indonesia/

A simple way to detect vacant parking spaces is to apply the background subtraction method. If you use the OpenCV library, you can learn this method with this tutorial: http://docs.opencv.org/3.2.0/d1/dc5/tutorial_background_subtraction.html.

The idea is that we capture all of the parking area without any vehicles. This will be used as a baseline background. Next, we can check whether a vehicle is present or not.

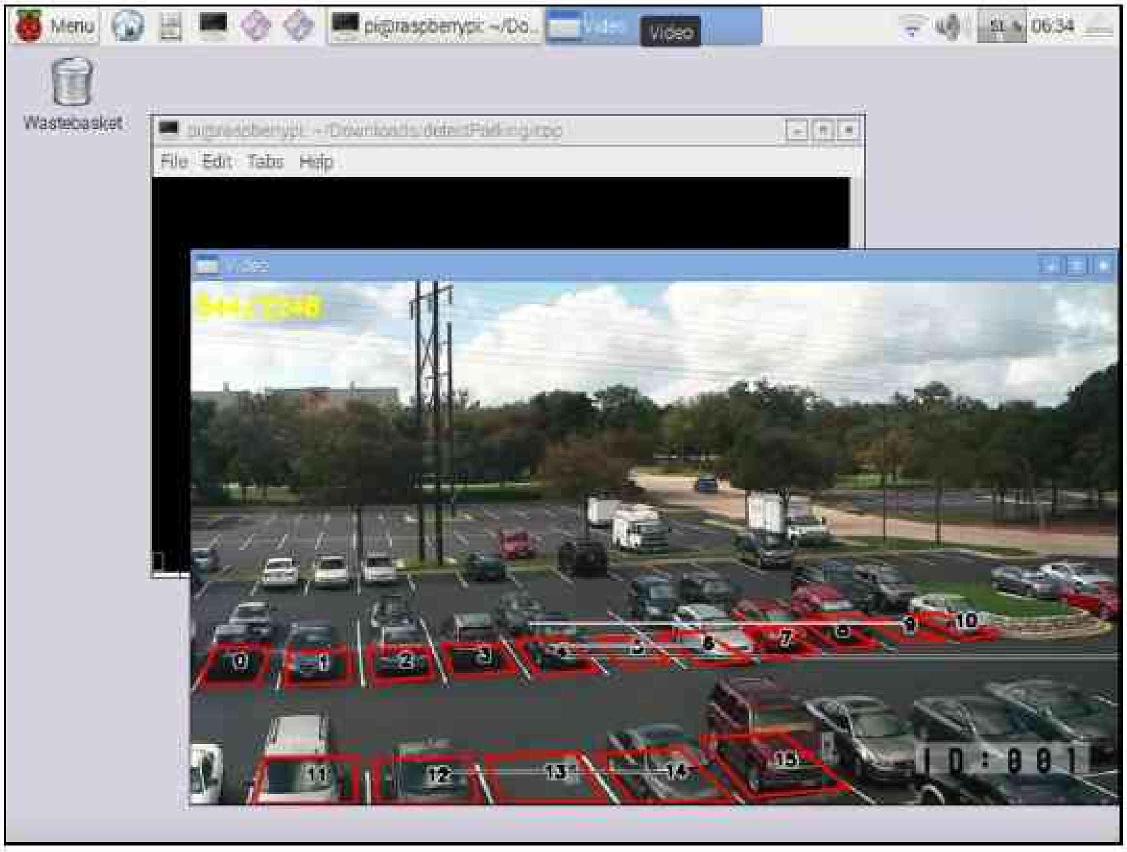


Source image: http://docs.opencv.org/3.2.0/d1/dc5/tutorial_background_subtraction.html

For another test, you can run a sample demo from https://github.com/eladj/detectParking. This application requires the OpenCV library. The application uses a template file to detect vehicle presence. For testing, you should run it on desktop mode. Open a terminal and run these commands:

```
$ git clone https://github.com/eladj/detectParkin
    $ cd detectParking/cpp/
    $ g++ -Wall -g -std=c++11 `pkg-config --cflags --libs opencv` ./*.cpp -o ./detect-p
    $ ./detect-parking ../datasets/parkinglot_1_480p.mp4 ../datasets/parkinglot_1.txt
```

You should see parking space detection as shown here:

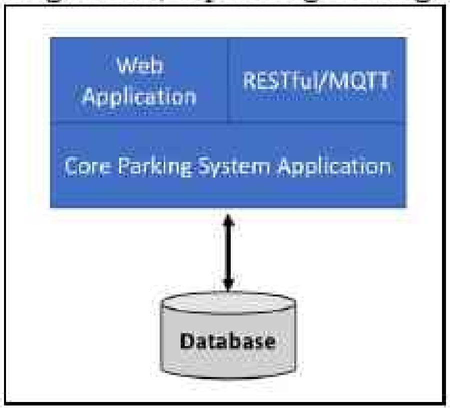


If you want to learn more about parking space detection and test your algorithm, you can use a dataset for testing from http://web.inf.ufpr.br/vri/news/parking-lot-database.

A parking management system

We have built sensors for detecting vehicle entry/exit and plate numbers. All those systems should be connected to a central system. This could be called a core system. To build a core system for parking, we could design and develop a parking management system. It's software which can use any platform, such as Java, .NET, Node.js, Python.

In general, a parking management system can be described as follows:

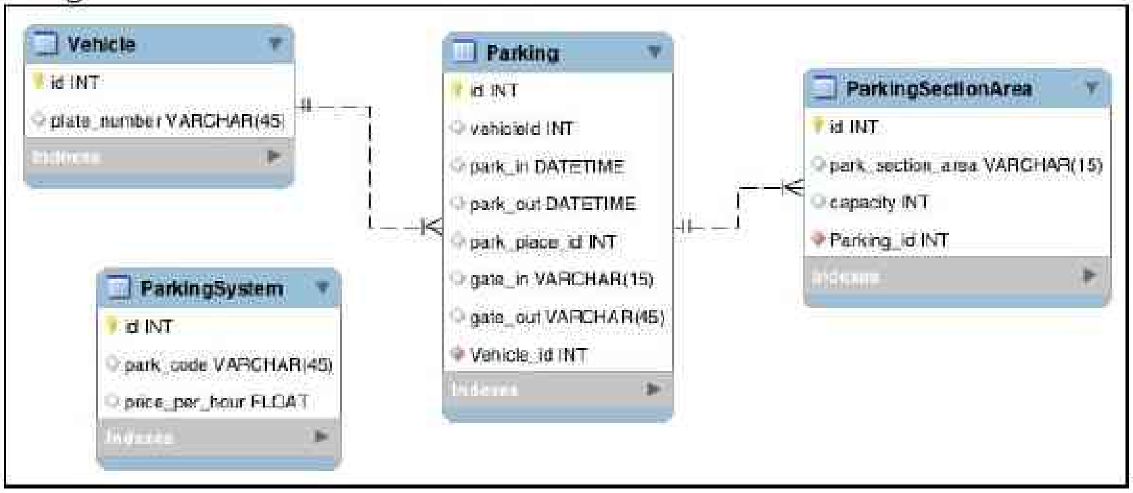


From this figure, we can explain each component as follows:

- **Core parking system application** is the main application for the parking system. This application is responsible for processing and querying data for all vehicles and sensors.
- **Web application** is a public application that serves parking information, such as capacity and current vacant parking space.
- **RESTful/MQTT** is an application interface to serve JSON or MQTT for mobile and sensor applications.
- **Database** is a database server that is used to store all data related to the parking application.

A database server is used to store all transactions in the parking system. We need to design a database schema based on our use case. Your database design should have plate number as transaction key. We'll record data for vehicle entry/exit. Lastly, you should design a parking pricing model. A simple pricing model is hourly--the vehicle will be charged at hourly intervals.

Now you can design your database. For demonstration, you can see my database design here:



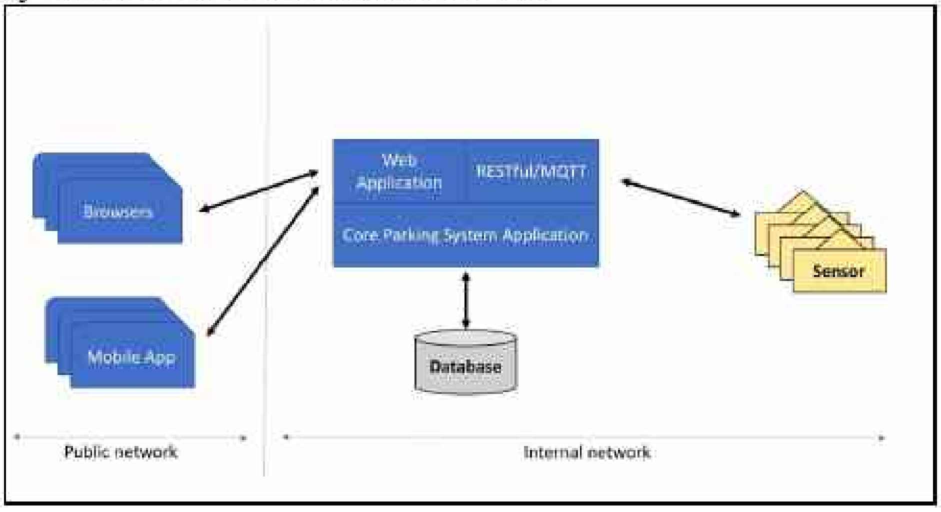
Here's an explanation of each table:

- **ParkingSystem** is general information about the current parking lot. You can set an identity for your parking lot. This is useful if you want to deploy many parking spaces at different locations.
- **Vehicle** is vehicle information. We should at least have plate numbers of vehicles.
- **ParkingSectionArea** is information about the parking section areas. It's used if you have many parking sub-areas in one parking location.
- **Parking** is a transaction table. This consists of vehicle information such as vehicle in/out time.

Again, this is a simple database design that may fit with your use case. You can extend it based on your problem.

Building a smart parking system

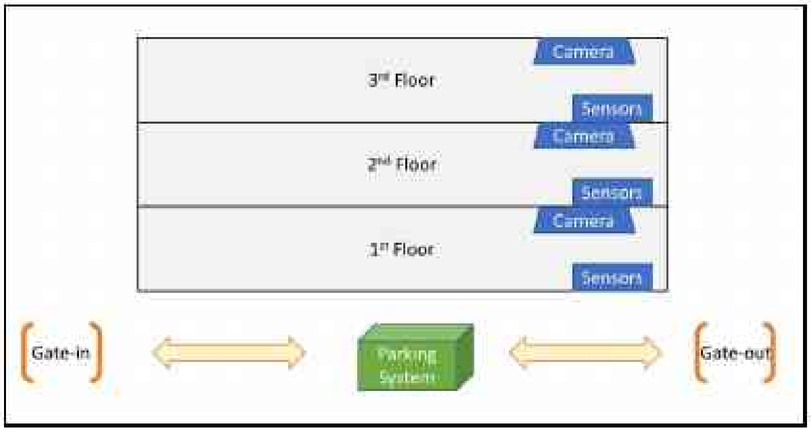
Building a smart parking system needs several components to make it work. These components can be software and hardware. In general, a smart parking system can be described as follows:



A parking application system consists of application modules that manage each task. For instance, the web application is designed as an interface for user interaction. All data will be saved into a storage server, such as MySQL, MariaDB, PostgreSQL, Oracle, or SQL Server.

All sensors are connected to a board system. You can use Arduino or Raspberry Pi. The sensor will detect vehicle entry/exit and plate numbers.

Firstly, we design a smart parking system in a building. We put a camera and sensors on each floor in the building. You also build gates for vehicle entry/exit with sensors. Lastly, we deploy a core parking system in the building. Building infrastructure is built to connect all sensors to the core system. You can see the infrastructure deployment here:


Camera
3rd Floor
Sensors
Camera
2nd Floor
Sensors
Camera
1st Floor
Sensors
Gate-in
Parking System
Gate-out

Summary

We learned what a smart parking system is and explored sensor devices that are used in a smart parking system. We also implemented core components of such a smart parking system.

In the next chapter, we'll review a vending machine and try to build a prototype.

Making Your Own Vending Machine

A vending machine is an automated machine that sells products such as snacks, beverages, cigarettes, and so on. Some vending machines sell tickets, puppets, and shirts. In this chapter, we will explore various vending machine technologies so you can build your own vending machine.

We will cover the following topics:

- Introducing vending machines
- Designing a vending machine
- Central control machine
- Detecting coins for payments
- Building UI and UX for user interaction
- Designing a database model
- Building the vending machine

Let's get started!

Introducing vending machines

In cities, you can find many vending machines at airports, bus terminals, offices, and campuses. Imagine you are at a bus terminal and you are thirsty. You can buy a beverage bottle directly from a vending machine. This is one of the scenarios of vending machines' operating environment.

A vending machine consists of various products. Users can select the product that they want to buy. After selecting the product, users pay by inserting coins or payment cards such as debit/credit cards. If payment is confirmed, the vending machine will deliver the selected product to the users.

You can see a vending machine in the following figure. You can check it out at https://www.samsclub.com/sams/seaga-16-snack-and-14-beverage-full-size-combo-machine/117240.ip:

In this chapter, we'll learn and explore how vending machines work and how to build one for your own targets and purposes.

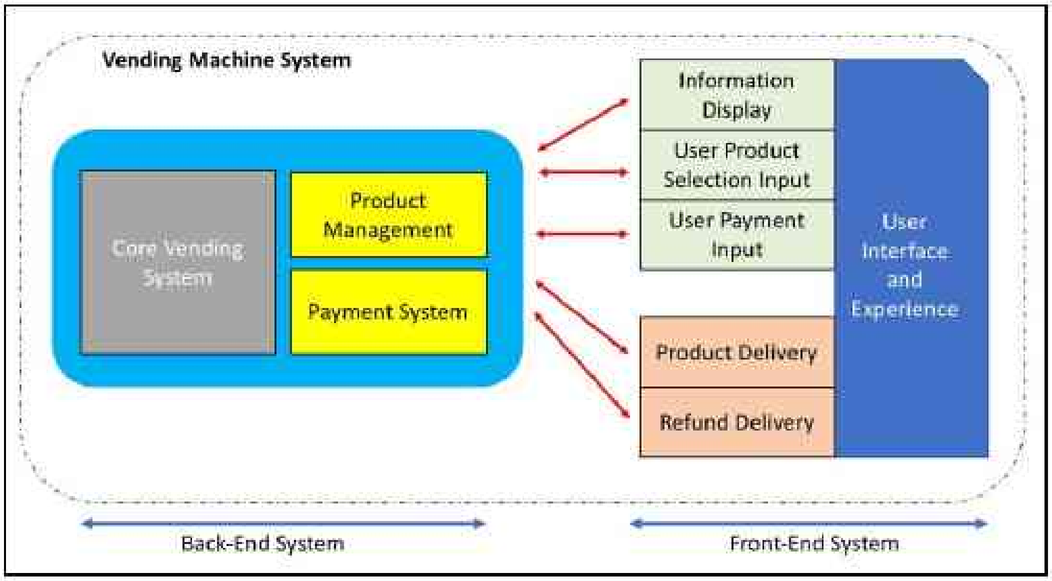
Designing a vending machine

Building a vending machine requires more attention to designing that usual. In general, we can design a vending machine system with the following parts:

- Machine cabinet
- Product storage
- Central processor machine
- Payment system
- User interface and experience

In this book, we won't explore machine cabinet and product storage for vending machines. You can review vending machine spare parts on this site: https://www.vendingworld.com/vending-machine-parts.

In general, a vending machine model can be described as follows:



A central processing machine can control all the processes in a vending machine. It can control a user's input and deliver the product. We can use Raspberry Pi and Arduino as a central processing machine since these boards provide GPIO for sensor and actuator devices.

Product management addresses product tracking and ensures that the product delivery is correct. In one scenario, if your vending machine is connected to the internet, the machine can inform the administrator about the current product stock.

Payment method is one of the problems in vending machines. We should be able to identify coin types and their value. This has potential problems of fraud. Some vending machines allow payment using debit/credit cards.

Central control machine

Most vending machine use FPGA to control all processes and transactions. One of the core controls of a vending machine is the AMS Sensit 2 PC board that you can see here:

For further information about this machine, you can go to https://www.vendingworld.com/ams-sensit-2-pc-board.php.

For a simple vending machine, you can use an Arduino board as the control unit in your vending machine. Arduino provides more digital and analog I/O pins that you can attach to the machine. You can extend these I/O pins using additional ICs.

Moreover, we can use your Raspberry Pi as a control unit for your vending machine. The Raspberry Pi has capabilities of a computer with low-end features. You can use desktop libraries such as OpenCV with Raspberry Pi to perform image processing and detection.

Detecting coins for payments

Most vending machines use coins as a payment method. Detecting coins is tricky. A potential problem is fraud. A coin may be actual money or a custom coin. Using custom coins, users can defraud the system.

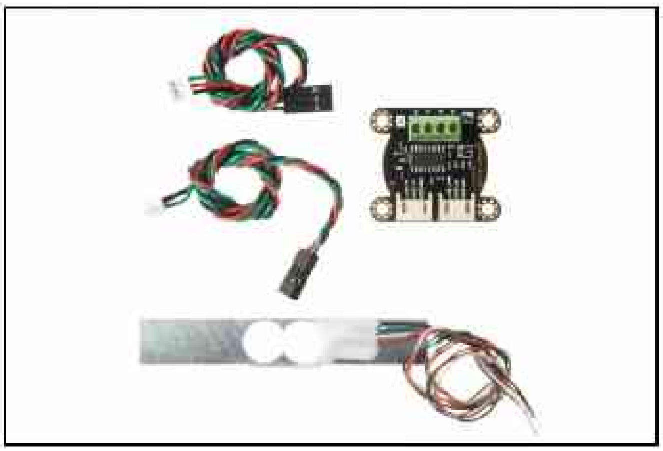
A sample of coins used in European countries can be seen here:

In this section, we'll explore two methods to detect coins. The first method is to detect by weight. The second method is to detect coins using optical sensors such as cameras. We'll explore these methods in the next section.

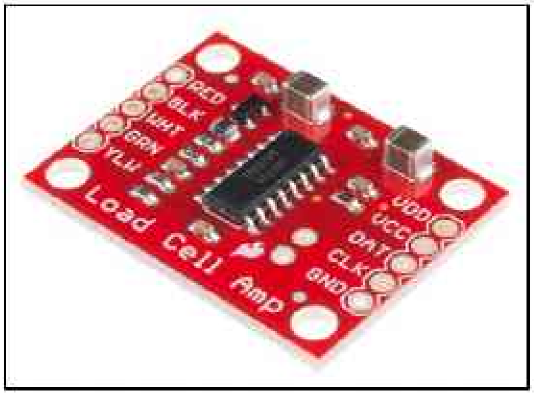
Detecting coin weight

In some cases, each coin has a unique weight. From this scenario, we can detect the kind of coin that the user enters into a vending machine. There are many weight sensors with various specific precision features.

The digital weight sensor Gravity, from DFRobot, is one weight sensor that you can use for your vending machine. You can buy this product from https://www.dfrobot.com/product-1031.html. This sensor is a complete kit so you can use it directly with your Arduino board. You can see this sensor here:



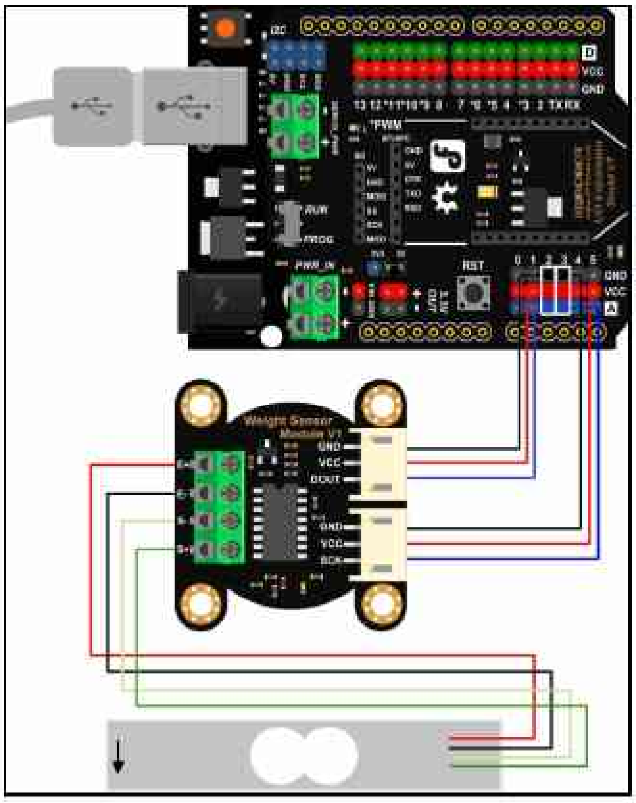
It uses an HX711 IC to measure object weight. You can buy the HX711 module and load cells of different models. For instance, you can use the HX711 module from SparkFun (https://www.sparkfun.com/products/13879) and a load cell from SparkFun (https://www.sparkfun.com/products/13329 and https://www.sparkfun.com/products/13332). You can see an HX711 module, called a Load Cell Amp, here:



For demonstration, we'll try to measure coin weight using DFRobot's Gravity sensor on an Arduino board. Based on its documentation, we can connect Gravity to Arduino through analog inputs with the following wiring:

- Sensor DOUT pin to Arduino A2 pin
- Sensor SCK pin to Arduino A3 pin

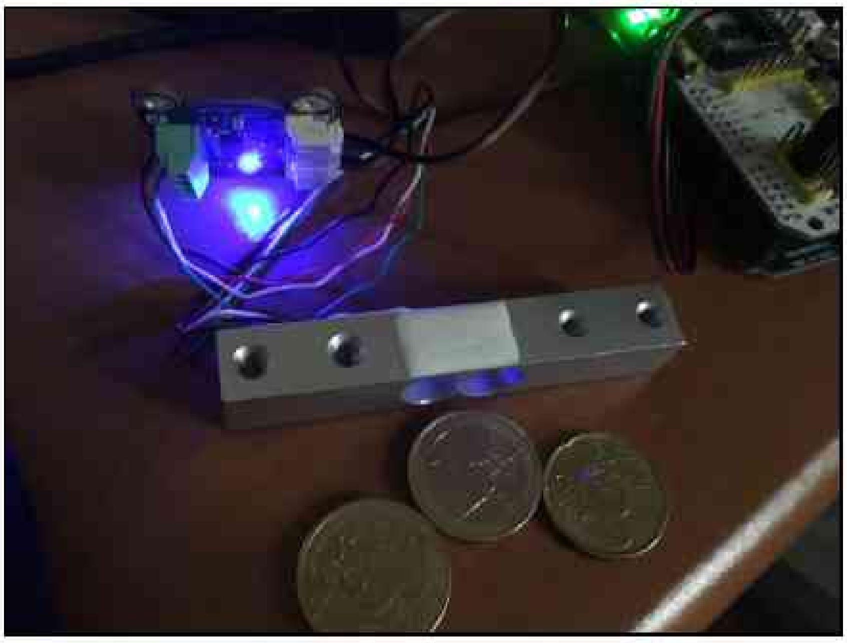
You can see the complete wiring here below:



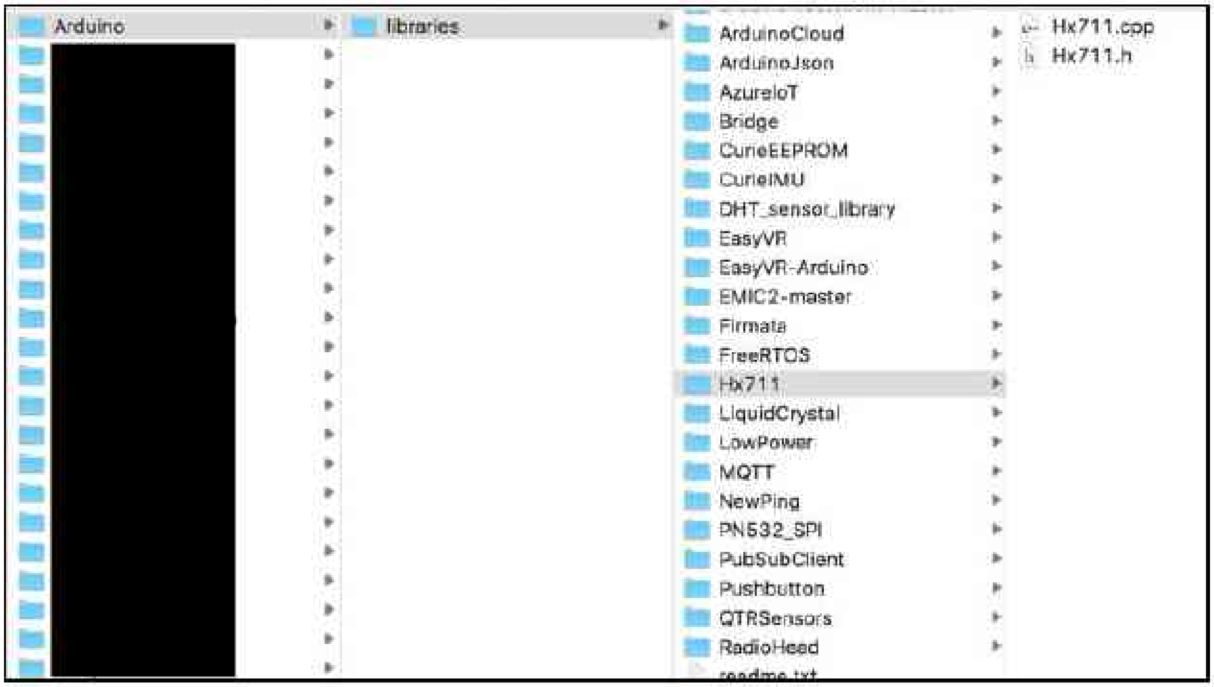
Source: image from DFRobot (http://www.dfrobot.com)

The wiring needs an I/O expansion shield for the Arduino (https://www.dfrobot.com/product-1009.html) in order for the digital weight sensor to be attached, but you can connect this weight sensor to the Arduino directly. Connect the sensor DOUT and SCK pins to Arduino Analog A2 and A3 pins.

After completing the hardware wiring, we need some coins for testing. In this case, I'll use three coins: 1 euro, 50 euro cents, and 20 euro cents. You can see my wiring and euro coin samples here:



Now we can write a sketch program to measure the coin weight. You can open your Arduino IDE to write the sketch program. To access the HX711 IC on Arduino, we need an additional library that you can download and extract from https://github.com/aguegu/ardulibs/tree/master/hx711. Put this library as the Hx711 folder inside the Arduino libraries folder. You can see the HX711 library on my computer here:



Arduino
[redacted]

libraries

ArduinoCloud
ArduinoJson
AzureIoT
Bridge
CurieEEPROM
CurieIMU
DHT_sensor_library
EasyVR
EasyVR-Arduino
EMIC2-master
Firmata
FreeRTOS
Hx711
LiquidCrystal
LowPower
MQTT
NewPing
PN532_SPI
PubSubClient
Pushbutton
QTRSensors
RadioHead
readme.txt

Hx711.cpp
Hx711.h

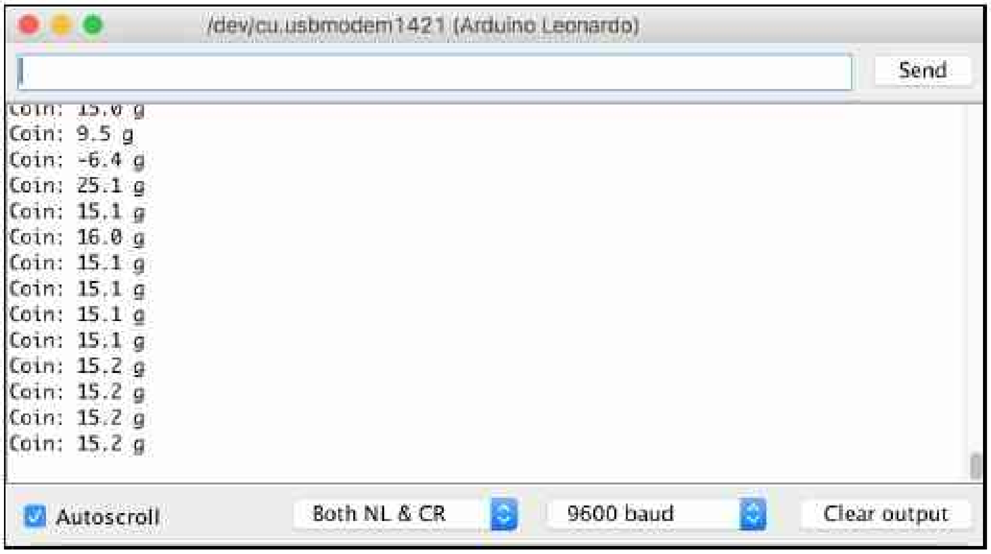
We develop a sketch program to measure object weight. We'll call it `getGram()` since it gets object weight in grams. Write the following program:

```
// Hx711.DOUT - pin #A2
// Hx711.SCK - pin #A3

#include <Hx711.h>
Hx711 scale(A2, A3);

void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.print("Coin: ");
  Serial.print(scale.getGram(), 1);
  Serial.println(" g");
  delay(500);
}
```

Save and upload the program to your Arduino board. Then you can open the Serial Monitor tool from Arduino to see the program output.

Make sure you put the load cell in the correct position. You can see my load cell here:

You can see the current object's weight in the Serial Monitor tool. At first, you'll see approximately 0 grams. Then, try putting a coin onto the load cell. You can note the total weight before and after measurement. You can also can perform a calibration on the sensor.



```
Coin: 15.0 g
Coin: 9.5 g
Coin: -6.4 g
Coin: 25.1 g
Coin: 15.1 g
Coin: 16.0 g
Coin: 15.1 g
Coin: 15.1 g
Coin: 15.1 g
Coin: 15.1 g
Coin: 15.2 g
Coin: 15.2 g
Coin: 15.2 g
Coin: 15.2 g
```

Based on my experiments with the digital weight sensor from DFRobot and Euro coins, I have the following results:

Coin model	Weight	Original weight
1 euro (€1)	6.7 grams	7.5 grams
50 euro cents (€0.50)	7.1 grams	7.8 grams
20 euro cents (€0.20)	4.5 grams	5.74 grams

To improve accuracy, we can make calibrations or change the measurement box. Since the sensor has a weight range of 1 kg, we may change a load cell with a low weight range to enhance our measurement.

Detecting coins using optical sensing

One of the common methods to detect a coin is using optical sensors such as cameras. We can identify various coin models. You can connect your camera to embedded systems such as Raspberry Pi. Then, you can perform image processing using OpenCV. We will try this on next. Please install OpenCV for your platform. Please read http://opencv.org for installation process.

For testing, we'll develop a Python application to detect coins. I'll use Euro coins for demonstration. The application will detect 1 euro. The easier method to detect coins is to implement template matching. We extract an image as a source for a template.

Then, we try to match that image:

To write a program, your platform should have OpenCV with Python bindings installed. Consider that we have a file, `1euro.jpg`, for template matching source and `coins.jpg` file for testing.

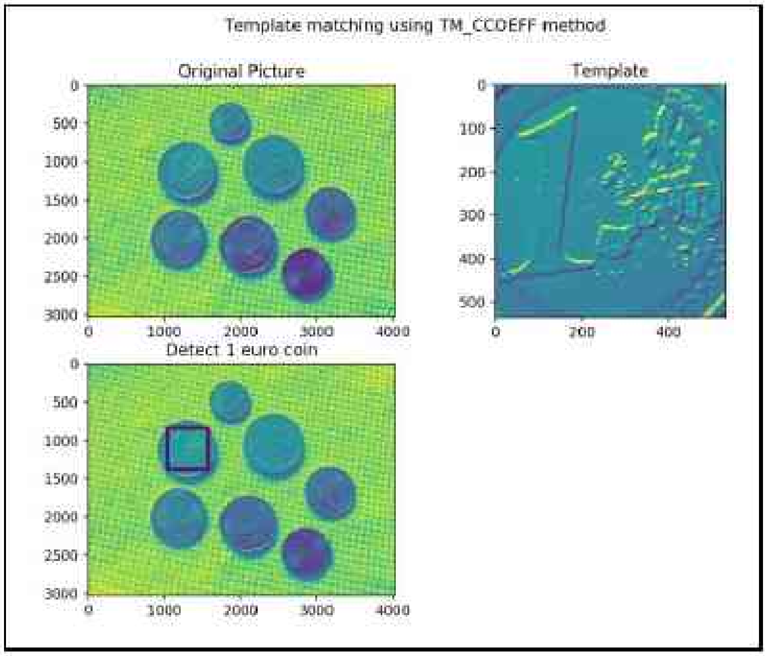
Now we'll write our Python program. You can write this script:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('coins.JPG',0)
img2 = img.copy()
template = cv2.imread('1euro.JPG',0)
img3 = template.copy()
w, h = template.shape[::-1]

# Apply template Matching
res = cv2.matchTemplate(img,template,cv2.TM_CCOEFF)
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)

top_left = max_loc
bottom_right = (top_left[0] + w, top_left[1] + h)
cv2.rectangle(img,top_left, bottom_right, (0,255,255), 50)

plt.subplot(221),plt.imshow(img2)
plt.title('Original Picture')
plt.subplot(222),plt.imshow(img3)
plt.title('Template')
plt.subplot(223),plt.imshow(img)
plt.title('Detect 1 euro coin')
plt.suptitle('Template matching using TM_CCOEFF method')

plt.show()
```

Save this script to a file called `coindetect.py`. After that, you can run this program by typing this command:

```
$ python coindetect.py
```

After executing the program, you should see a dialog that shows that a 1-euro coin is detected. You can see a sample program output here:



1. Firstly, we load the template and tested files.

```
img = cv2.imread('coins.JPG',0)
img2 = img.copy()
template = cv2.imread('1euro.JPG',0)
img3 = template.copy()
w, h = template.shape[::-1]
```

2. Then, we apply our template matching method by calling `matchTemplate()` with the `TM_CCOEFF` parameter. For further information about this function, read the OpenCV documentation at http://docs.opencv.org/3.0-beta/doc/tutorials/imgproc/histograms/template_matching/template_matching.html:

```
res = cv2.matchTemplate(img,template,cv2.TM_CCOEFF)
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
```

3. If a match is found to the template file on the target image, we draw a rectangle on the image:

```
top_left = max_loc
bottom_right = (top_left[0] + w, top_left[1] + h)
cv2.rectangle(img,top_left, bottom_right, (0,255,255), 50)
```

4. Lastly, we plot all results of our computation:

```
plt.subplot(221),plt.imshow(img2)
plt.title('Original Picture')
plt.subplot(222),plt.imshow(img3)
plt.title('Template')
plt.subplot(223),plt.imshow(img)
plt.title('Detect 1 euro coin')
plt.suptitle('Template matching using TM_CCOEFF method')
plt.show()
```

For your experiments, you can modify parameters of `matchTemplate()` to get more accuracy.

The `template-matching` method has a disadvantage. If a coin is rotated, this method cannot detect it. You should rotate the coin image to fit your template source.

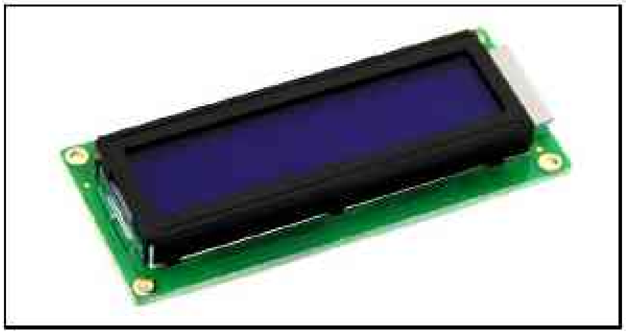
Building UI and UX for user interaction

UI (user interface) and UX (user experience) are two parameters that are used for user interaction. A user can select products, pay, and take selected products. This is the normal flow in a vending machine.

In this section, we'll explore some UI and UX for vending machine.

Display interfacing

A simple display that can be used with Arduino or Raspberry Pi is an LCD of 16x2 characters. It consists of 16 characters in 2 lines. This LCD module is a low-cost display and easy to use. You can see the form of the LCD 16x2 display here:

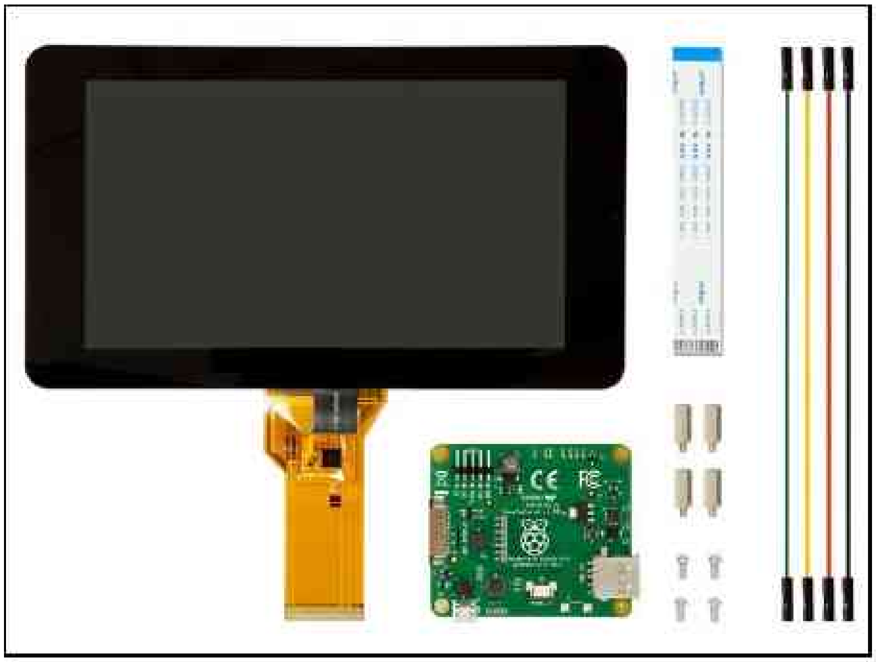


If you use LCD 16x2 characters with an Arduino, we can use the LiquidCrystal library. You can read about it at https://www.arduino.cc/en/Reference/LiquidCrystal. I recommend you test with the `Hello World` tutorial from Arduino on this site: https://www.arduino.cc/en/Tutorial/HelloWorld:

```
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("vending machine!");
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```

Compile and upload this sketch to Arduino. You should see vending machine! on the LCD 16x2 characters module.

Currently, display technology is grow in fast. Touchscreen displays can be used for your vending machine design. For instance, if your core system in the vending machine uses Raspberry Pi, you can use the official Raspberry Pi 7" touchscreen display. You can buy it on https://www.raspberrypi.org/products/raspberry-pi-touch-display/. You can see this display model here:



Follow the assembly instructions for the display from the datasheet document, which included in the box. You can use this assembly file from Adafruit: https://cdn-shop.adafruit.com/product-files/2718/2718build.jpg.

After constructing the display, you should upgrade and install some libraries on Raspberry Pi's Raspbian OS. You can type these commands:

```
$ sudo apt-get update
    $ sudo apt-get upgrade
    $ sudo apt-get dist-upgrade
    $ sudo apt-get install raspberrypi-ui-mods
    $ sudo apt-get install raspberrypi-net-mods
```

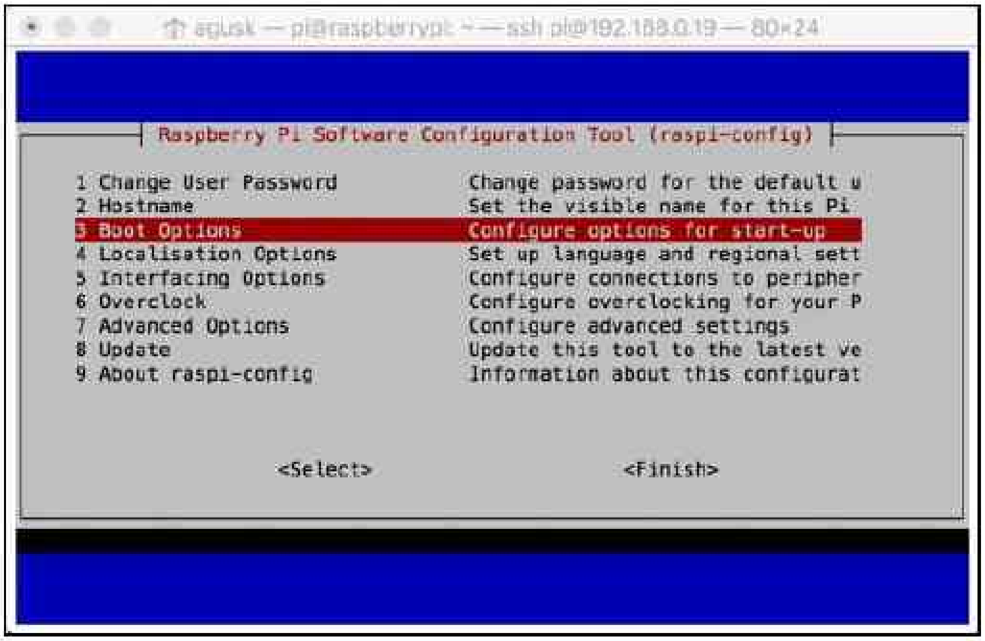
Now you can work with the Raspberry Pi 7" touchscreen display like on a tablet.

You can use a UI on Raspberry Pi using a browser. You can create a web application and then that browser inside Raspberry Pi will call it.

Make sure your Raspberry Pi is started in GUI mode. You can configure this via the `raspi-config` tool. You can call it by typing this command:

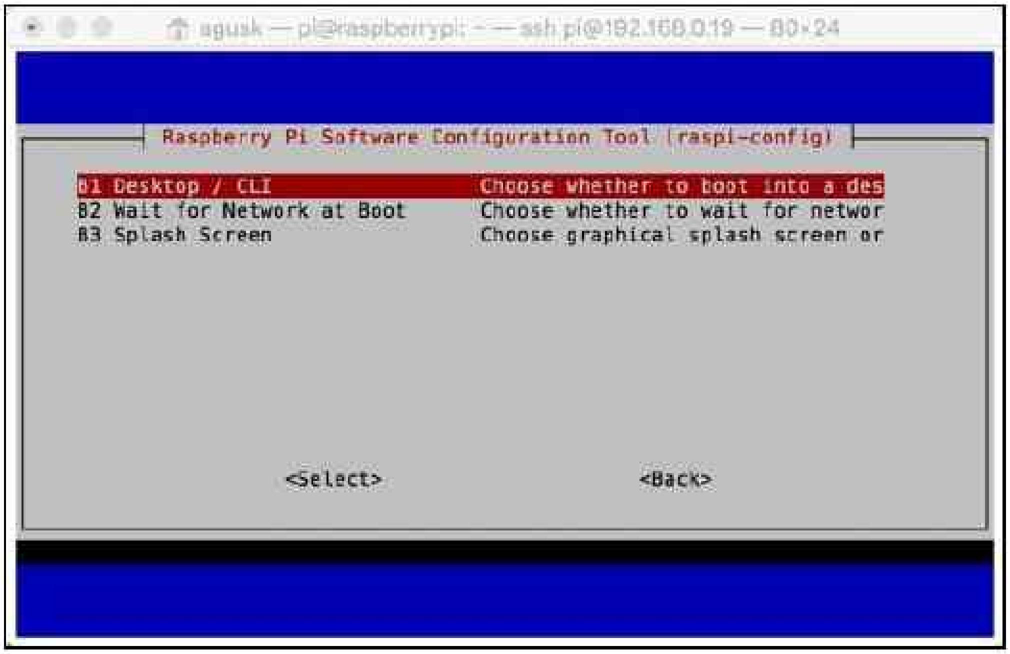
```
$ sudo raspi-config
```

After it has executed, you should see this menu:



```
agusk — pi@raspberrypi: ~ — ssh pi@192.168.0.19 — 80×24

Raspberry Pi Software Configuration Tool (raspi-config)

1 Change User Password        Change password for the default u
2 Hostname                    Set the visible name for this Pi
3 Boot Options                Configure options for start-up
4 Localisation Options        Set up language and regional sett
5 Interfacing Options         Configure connections to peripher
6 Overclock                   Configure overclocking for your P
7 Advanced Options            Configure advanced settings
8 Update                      Update this tool to the latest ve
9 About raspi-config          Information about this configurat

              <Select>                 <Finish>
```

Select 3 `Boot Options` so you have options to select Desktop/CLI for starting on Raspbian boot:



```
agusk — pi@raspberrypi: ~ — ssh pi@192.168.0.19 — 80×24

Raspberry Pi Software Configuration Tool (raspi-config)

B1 Desktop / CLI               Choose whether to boot into a des
B2 Wait for Network at Boot    Choose whether to wait for networ
B3 Splash Screen               Choose graphical splash screen or

              <Select>                 <Back>
```

Let's assume we use the Chromium browser on Raspbian Jessie to run our web application. You can install it on Raspberry Pi:

```
$ sudo apt-get update
    $ sudo apt-get upgrade
    $ sudo apt-get install unclutter
```

Then, we configure `autostart` on Raspberry Pi. Type this command:

```
$ nano ~/.config/lxsession/LXDE-pi/autostart
```

You can modify this script:

```
@lxpanel --profile LXDE-pi
@pcmanfm --desktop --profile LXDE-pi
@xset s off
@xset -dpms
@xset s noblank
@sed -i 's/"exited_cleanly": false/"exited_cleanly": true/' ~/.config/chromium-browser
@chromium-browser --noerrdialogs --kiosk [URL] --incognito --disable-translate
```

Change `[URL]` to the `URL` of your web application. It can be a local web or remote web application.

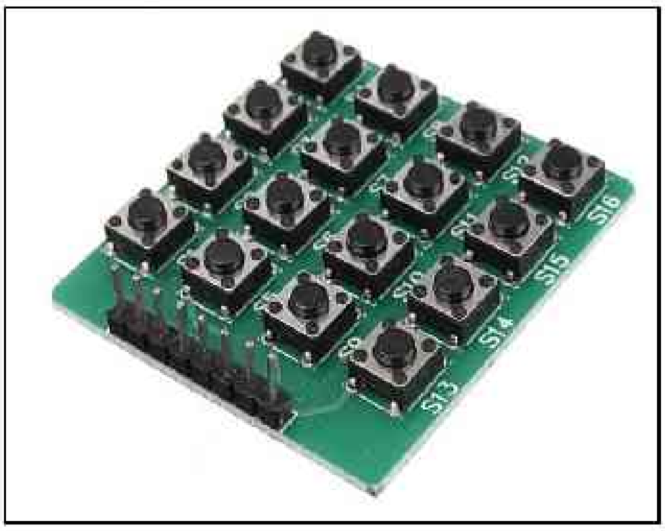
Now you can reboot your Raspberry Pi. Once rebooting is complete, you should see the browser open your web application.

Input interfacing

An input interface is a module that is used to get input from users. Input interfacing is applied to vending machines in order to have user interaction. Users can select a product that they want to buy. Each product has labels so users only select a product from the input interface by pressing the product label.

A simple input module is the Membrane 3x4 Matrix Keypad from Adafruit: https://www.adafruit.com/product/419. This module provides 3x4 inputs that Arduino or Raspberry Pi can read. You can see the Membrane 3x4 Matrix Keypad here:

I also found another input from Banggood. It's a 4x4 Matrix Keypad that is made with buttons. You can see it here:.



To use this keypad, we should know how many keys are there on the keypad. If we implement it in Arduino, we can use Keypad Library for Arduino: http://playground.arduino.cc/Code/Keypad. You can download this library and put it in the Arduino libraries folder.

Consider we use a 3x4 matrix keypad module to attach to the Arduino board. You can write this sketch program:

```
#include <Keypad.h>

const byte ROWS = 4; //four rows
const byte COLS = 3; //three columns
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'#','0','*'}
};
//connect to the row pinouts of the keypad
byte rowPins[ROWS] = {5, 4, 3, 2};
//connect to the column pinouts of the keypad
byte colPins[COLS] = {8, 7, 6};

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

void setup(){
  Serial.begin(9600);
}

void loop(){
  char key = keypad.getKey();

  if (key != NO_KEY){
    Serial.println(key);
  }
}
```

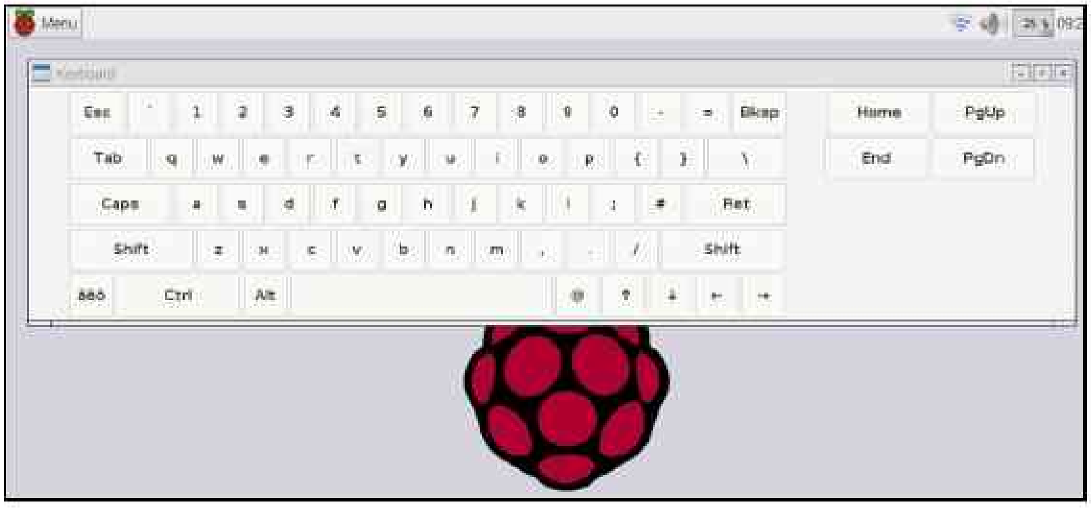
Compile and upload this program to your Arduino board.

Now you can open the Serial Monitor tool from Arduino to see the program output. You can press any keypad on the module in order to see the program output.

If you use the Raspberry Pi 7" touchscreen display, we can use matchbox keyboard software to enter data. You can install it by typing these commands:

```
$ sudo apt-get update
    $ sudo apt-get upgrade
    $ sudo apt-get install matchbox-keyboard
```

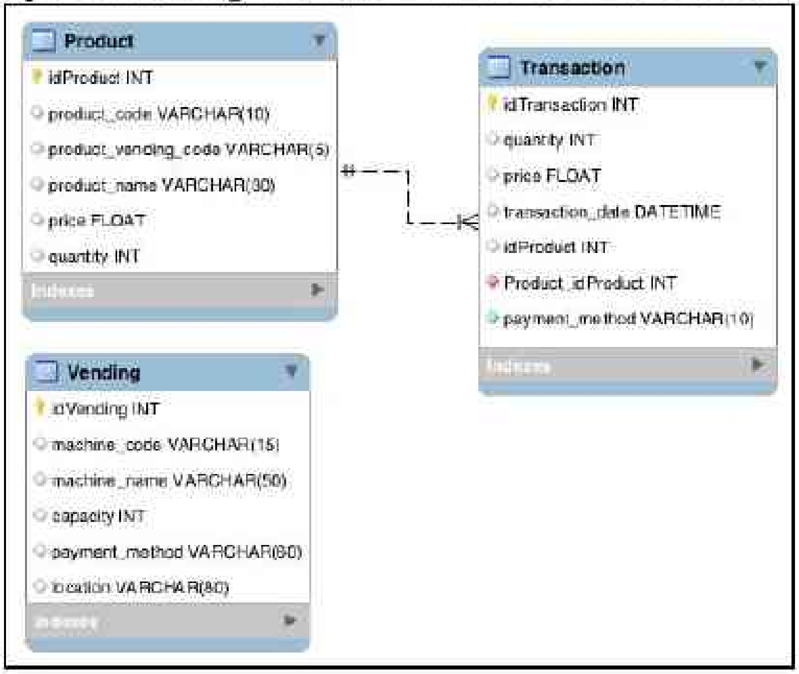
If done, you can reboot your Raspberry Pi. Now you can see the matchbox keyboard. You can find this keyboard from `MENU >> ACCESSORIES >> KEYBOARD.`


Menu
Keyboard
Esc ` 1 2 3 4 5 6 7 8 9 0 - = Bksp Home PgUp
Tab q w e r t y u i o p [ ] \ End PgDn
Caps a s d f g h j k l ; # Ret
Shift z x c v b n m , . / Shift
äëö Ctrl Alt @ ↑ ↓ ← →

Designing a database model

If you have a vending machine with various products and payment methods, it's recommended you use a database to store all transactions in the vending machine.

Three data points that you should take care about are product, transaction, and machine vending capabilities. In general, we can describe the core database design as follows:



The `Product` table consists of product information, including quantity and price. If you look at the schema in the figure, there are two fields: `product_code` and `product_vending_code`. `product_code` is the internal manufacturing identity that the system will use for all transactions. You can put **Stock Keeping Unit** (**SKU**) data into this field. Each product usually has an SKU ID so we can use it for our product identity. Otherwise, the `product_vending_code` field is designed for identity numbers on vending machines. These are the number that users see and select to buy on the vending machine. They're usually two- to five-digit numbers depending on how many products are in the vending machine.

The `Transaction` table is used for storing all transactions that occur in the vending machine. This is useful for logging and tracking. If the vending machine is connected to the internet, we can identify the current stock of each product remotely. Once product stock is low, we can deliver new products to the vending machine.

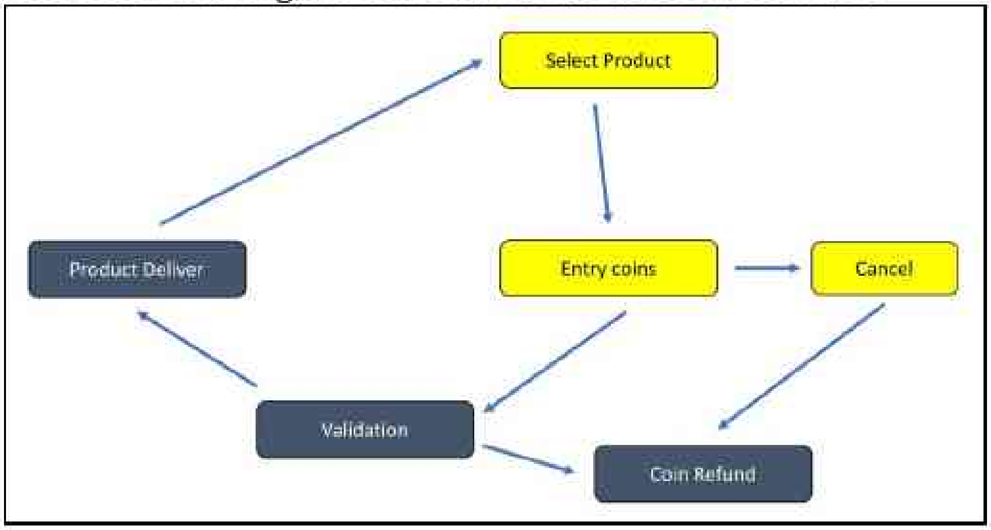
Lastly, the `Vending` table keeps information about machine capabilities such as product capacity, payment method, location, and so on. You can put all general information about the vending machine here.

This is a simple database design. You can extend it based on your use case and target while building your vending machine.

Building the vending machine

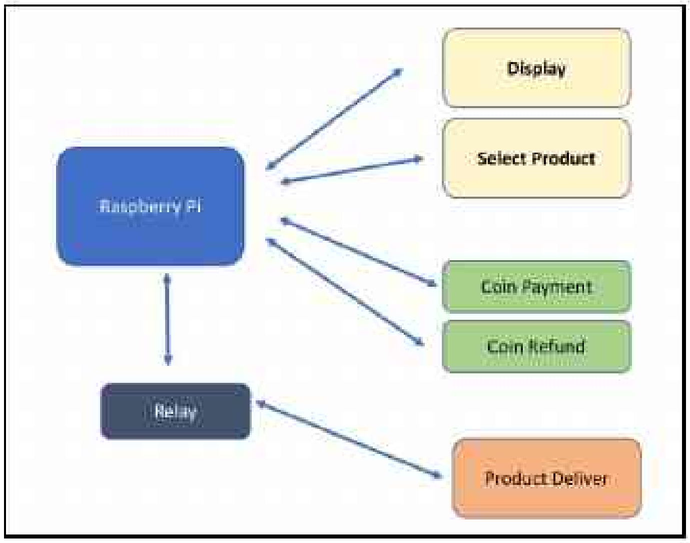
Building a vending machine is a process of integrating and combining several aspects of technology. Starting to build a cabinet for vending machine and designing coin and card payment method. This book does not explain to build a cabinet. Please try to look for information about building a cabinet for vending machine.

Technically, a vending machine is a **finite state machine** (**FSM**). The machine works under its states. Each state has been defined for input and output. A simple FSM for a vending machine can be described as follows:



Start with the **Select Product** state. It waits for input from the user. Once it is selected, the user can enter coins for the selected product. If the user cancels the transaction, the money will be refunded. If not, the system will validate the coins and compare it to the product price. If it's valid, the system will deliver the selected product.

In our implementation, we can use a Raspberry Pi as the central unit for the vending machine. The Raspberry Pi will control input from the user and handle product selection processing. It usually uses a motor, so we need relay modules to communicate with them.



The advantage of using a Raspberry Pi for vending machine implementation is to get the rich features it offers in computing and IoT capabilities. You can make a cluster of vending machines that can be controlled remotely.

Summary

We learned about designing a vending machine and its technology in this chapter. Various core technological aspects were also explored in order to get insight on how to build a vending machine, such as detecting coins and building UI and UX for user interaction.

In the next chapter, we will explore smart digital advertising and the technology that is used to build such a system.

A Smart Digital Advertising Dashboard

Every product and service should be marketed in order to get maximum sales. In this chapter, we'll learn to explore digital signage systems, including smart systems. We'll starting with reviewing existing digital signage platforms and go on to designing a smart digital advertising system and adding sensor devices to it.

We'll learn the following topics:

- Introducing smart digital advertising dashboards
- Exploring digital signage platforms
- Designing a smart digital advertising system
- Detecting human presence
- Displaying and delivering ad content
- Building a smart digital advertising dashboard

Let's get started!

Introducing smart digital advertising dashboards

While staying in a hotel or working in an office building, you may see a monitor with information. The content could be information about the building or advertising content such as product information.

For instance, look at the monitor with advertising content near the lift in the following image. You may see one in your office or mall building.

Source image: https://onelan.com/case-studies/corporate/pwc-a-communication-tool-to-reach-2000-employees.html

Another example you may see is the advertising monitors in airports on the walls:

Source image: http://www.digitalsignage.com.au/dsn/markets/airports.php

Products and services need to be marketed so people can know about them. It's important because if people need stuff or services that we are selling, they will contact us to get further information. It's an opportunity!

In this chapter, we'll learn to explore digital signage platforms to deliver ad content such as text, image, and video. We'll add features to the digital signage platform with specific intelligence to get optimized marketing programs.

Exploring digital signage platforms

Building a digital signage system for advertising needs more knowledge in implementing the system. Some companies and communities have built good digital signage platforms. In this section, we will review some platforms that can be deployed on the Raspberry Pi.

1Play

1Play is a digital signage system based on the Raspberry Pi (Pi Zero and Pi 1,2, and 3). It supports images, videos, RTSP video streaming, and HTML5 content. The content is pushed from a cloud server. You can also manage all content from a system. This platform is not free but you can try 1Play with a 30-day free trial. If you're interested, you can visit the official website at https://1play.tv/.

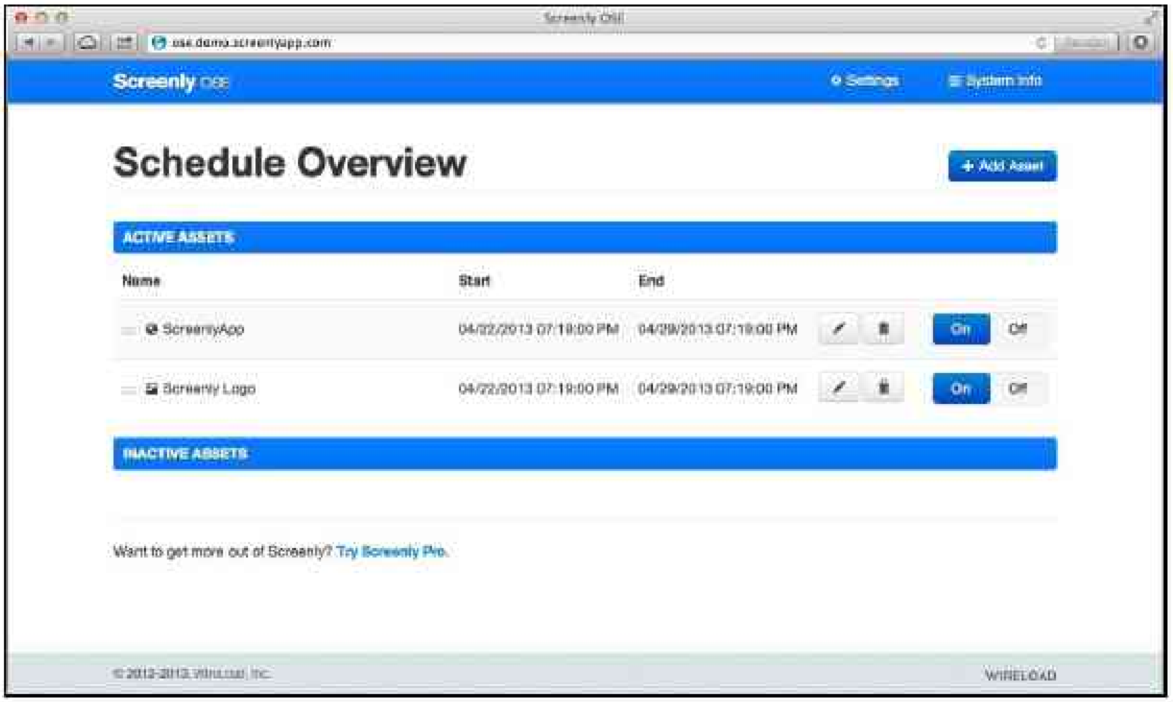
Screenly

Screenly is a digital signage platform-based web application. They provide commercial and open source editions. All ad content such as image, video, and links can be managed easily. Their official website is https://www.screenly.io. For the open source edition, you can read the installation instructions on https://www.screenly.io/ose/.

With respect to the Screenly open source edition, you can deploy it with a custom image that is provided by Screenly. You can download it from https://github.com/screenly/screenly-ose/releases. If you have a Raspberry Pi with Raspbian Jessie, you can install it directly into your Raspberry Pi. Just open terminal and type this command:

```
$ bash <(curl -sL https://www.screenly.io/install-ose.sh)
```

After it's installed and rebooted, you can see the Screenly dashboard. You can manage ad content by calling the URL `http://<server>:8080` on your remote web server. Don't run the web browser on your local Raspberry Pi. You can see the Screenly management page here:



Schedule Overview

ACTIVE ASSETS

Name	Start	End
ScreenlyApp	04/22/2013 07:19:00 PM	04/29/2013 07:19:00 PM
Screenly Logo	04/22/2013 07:19:00 PM	04/29/2013 07:19:00 PM

INACTIVE ASSETS

Want to get more out of Screenly? Try Screenly Pro.

© 2012-2013 Wireload, Inc.

Source image: https://www.screenly.io/ose/

Screenly is built using Python and runs on a web server called nginx. You can modify it for your use case. For further information about Screenly's open source edition, go to https://github.com/screenly/screenly-ose.

Xibo

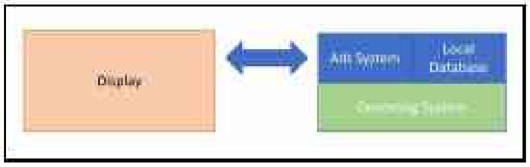
Xibo is an open source digital signage system. It provides ad content such as videos, images, RSS, text, clocks, tabular data, and so on. You can manage a schedule for ad content via a CMS portal. Xibo uses a PHP web application to deliver content and MySQL as database storage. For further information about Xibo, you can download and install it on http://xibo.org.uk.

Concerto

Concerto is a web-based digital signage system. Using Concerto, we can build digital signage systems to deliver ad content on the Raspberry Pi. It uses MySQL, SQLite, and Postgres for database options. It also provides a CMS system to manage your ad content. The official website of Concerto can be found at http://www.concerto-signage.org.

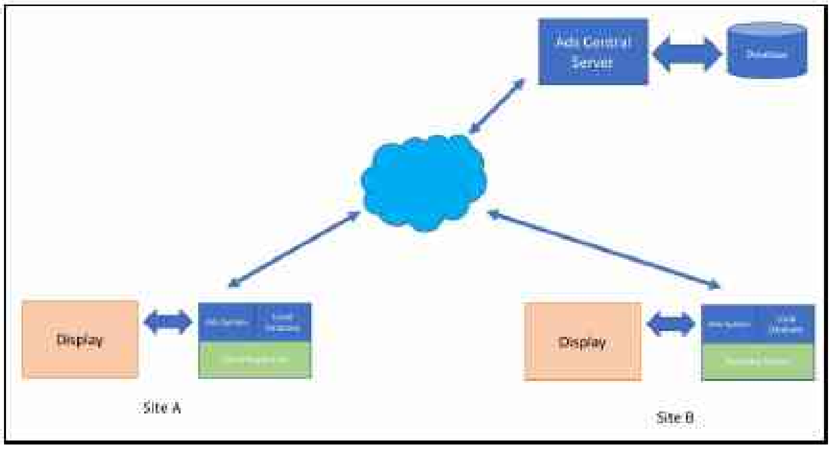
Designing a smart digital advertising system

A key part of building a smart digital advertising system is to deliver ad content and get insight. In general, we can build a digital advertising dashboard or digital signage system using a desktop or web application. This application will act as content host to deliver ad content. You can see a simple architecture of a digital signage system here:



The **Ads System** can be a desktop or web application. There is a local database to store ad content. The application will show ad content periodically.

In some cases, we can deploy multiple ad servers at several locations. We'll build a central database so ad content can be pushed to each local ad server. We also can design to deliver ad content based on location. It means each local ad server has a different ad location. In general, we can build a hybrid centralization and decentralization model as shown in this figure:

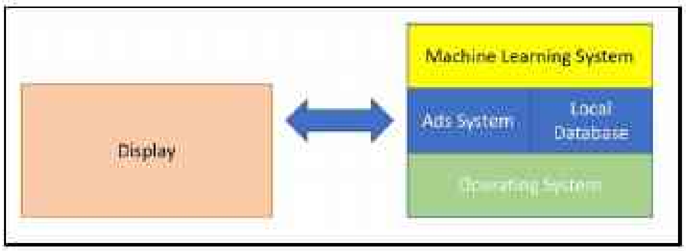


You can see from this figure that we need internet to connect between the server and local ad servers. Depending on your ad content type and size, we should estimate network bandwidth capacity in order to deliver ad content well. Ads content can be pushed to the local server at certain periods.

To make our digital signage system smarter, we can add intelligence features, for instance, detecting human presence while ad content is showing. The system should record that information and store it into a local server.

Getting information about the number of people watching ad content is necessary. We can record this information into a database. From that, we can get insight in order to determine the peak season for showing ad content.

In general, we can add a machine learning algorithm into the sensor device to obtain the density of human presence. You can see this in the following figure of a general architecture:



In a real-life implementation, we can use any sensor, including optical sensors like cameras, to detect human presence. We will discuss this in the next section.

Detecting human presence

In this section, we will explore some methods of detecting human presence. It's useful to deliver optimized ad content with information about the number of people watching a monitor. We can configure TV/monitor to go to sleep when there are no people in the area.

PIR motion sensor

A **Passive Infrared Detection** (**PIR**) motion sensor is used to detect object movement. We can use this sensor to detect human presence. You can find PIR sensors at SeeedStudio (https://www.seeedstudio.com/PIR-Motion-Sensor-Large-Lens-version-p-1976.html), Adafruit (https://www.adafruit.com/product/189), and SparkFun (https://www.sparkfun.com/products/13285). A PIR motion sensor usually has three pins: VCC, GND, and OUT. The OUT pin is the digital output; if we get value 1, it means motion is detected.

You can see the PIR sensor from SeeedStudio here:

There's another PIR motion sensor from SparkFun, the SparkFun OpenPIR: https://www.sparkfun.com/products/13968. This sensor provides analog output so that we can adjust the motion-detection value. Here's how it looks:

We learned how to use this sensor in Chapter 2, *A Smart Parking System*. You can try using it to detect human presence.

Ultrasonic sensor - HC-SR04

The HC-SR04 is a cheap ultrasonic sensor. It is used to measure the range between itself and an object. Each HC-SR04 module includes an ultrasonic transmitter, a receiver, and a control circuit. You can use this sensor to detect human presence in a range of 2 cm to 5 cm, depending on sensor accuracy.

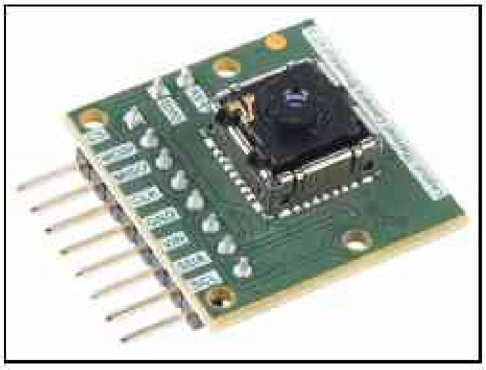
In general, the HC-SR04 module has four pins: GND, VCC, Trigger, and Echo. You can see the HC-SR04 from SparkFun (https://www.sparkfun.com/products/13959) in the following image. You can also find this sensor on SeeedStudio: https://www.seeedstudio.com/Ultra-Sonic-range-measurement-module-p-626.html. To save costs, you can buy the HC-SR04 sensor from Aliexpress. To learn how to use this sensor, refer to Chapter 2, *A Smart Parking System*.

Thermal sensor

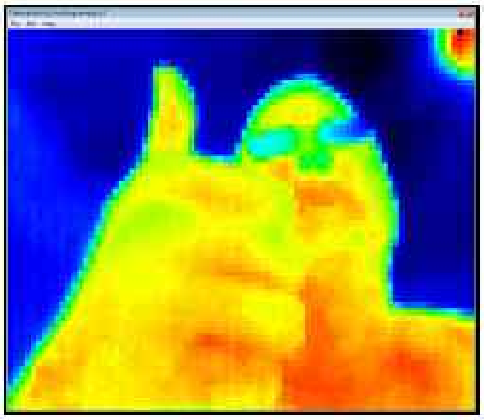
Humans can be detected by the surface temperature of their bodies. One sensor device that can be used to detect human presence based on temperature is the D6T MEMS thermal sensor: https://www.omron.com/ecb/products/sensor/11/d6t.html.

You can connect this sensor to an Arduino board or Raspberry Pi. It delivers sensor output through the I2C protocol. Here's how it looks:

The FLiR Dev Kit is a thermal imaging sensor from Pure Engineering that can be attached to Arduino and Raspberry Pi boards. The product can be reviewed at http://www.pureengineering.com/projects/lepton and is shown in the next image. You can also find it on SparkFun: https://www.sparkfun.com/products/13233:



Technically, this sensor will a generate thermal imaging photo. If you see red pixels in the photo, it means that area has a high temperature:

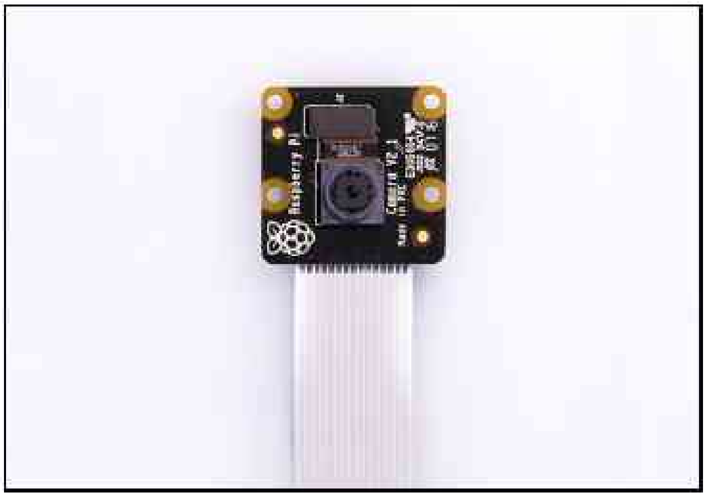


Source image: http://www.pureengineering.com/projects/lepton

Optical sensor

A camera can be used as an optical sensor to capture what's happening in a certain area. Every camera has different features. You should choose the appropriate camera for your use case. By adding artificial intelligence (AI) program with your camera, you can use it to detect human presence.

The Raspberry Pi Foundation provides the official camera for the Raspberry Pi. Currently, the recommended camera is the camera module version 2. You can read about and buy it at https://www.raspberrypi.org/products/camera-module-v2/. The Raspberry Pi foundation also has the Pi Camera Noire V2. This camera can work in low light. You can find this product at https://www.raspberrypi.org/products/pi-noir-camera-v2/. You can see it here:



A simple method to detect human presence is to use a face detection algorithm. When we connect a camera to an ad monitor, we can assume that if people are watching the monitor, their faces will be seen.

We can use OpenCV to implement face detection. For instance, we can detect a face in a picture file using Haar Cascade. You can use this program:

```
import numpy as np
import cv2


face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')

```

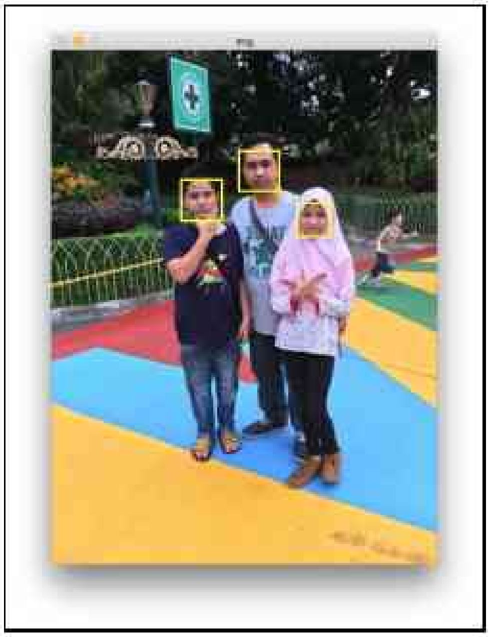
```
img = cv2.imread('IMG_FACE.JPG')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(gray, 1.3, 5)
for (x, y, w, h) in faces:
    img = cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 255), 2)

print "Number of faces detected: " + str(faces.shape[0])

cv2.imshow('img', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

You should change the image file with your test picture file. Save this program as `ch04_faces.py`.

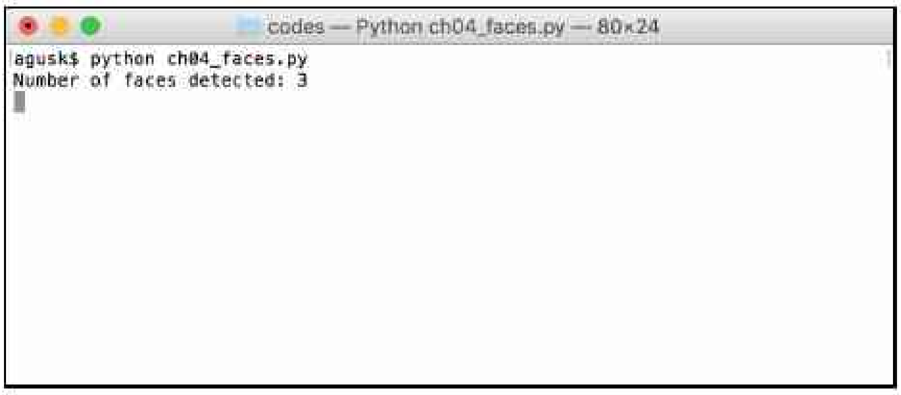
Now you can run the program on a Terminal. Type this command:

```
$ python ch04_faces.py
```

You can see rectangles on the picture if there is a human face. Take a look at the test picture here:



In a terminal, the program will display a message that shows the number of detected faces. You can see it here:



```
agusk$ python ch04_faces.py
Number of faces detected: 3
```

Displaying and delivering ad content

If you design advertising with a standalone model, you should have a display such as a monitor or a tablet in order to show ad contents. The Raspberry Pi has an official display of 7 inches, shown in the following image. This display can be attached to a particular box or wall.

You can also use a normal monitor to display ad contents. It's easier to attach to a Raspberry Pi through HDMI.

In this section, we'll review various backend technologies to deliver ad content. We'll focus on the backend technology stack with the Python platform. We'll explore web frameworks for Python that we can use to implement an ad server.

Flask

Flask is a simple web framework based on Werkzeug. It's easy to build `HTTP GET`/`POST` requests. If you have a scenario where you want to develop a RESTful server, Flask can solve your problem fast. The official website of Flask is http://flask.pocoo.org. You can install Flask using pip. Type this command:

```
$ pip install Flask
```

For testing, we can create a simple-to-handle HTTP GET request. Create a Python file, called `ch04_flask.py`. Write the following program:

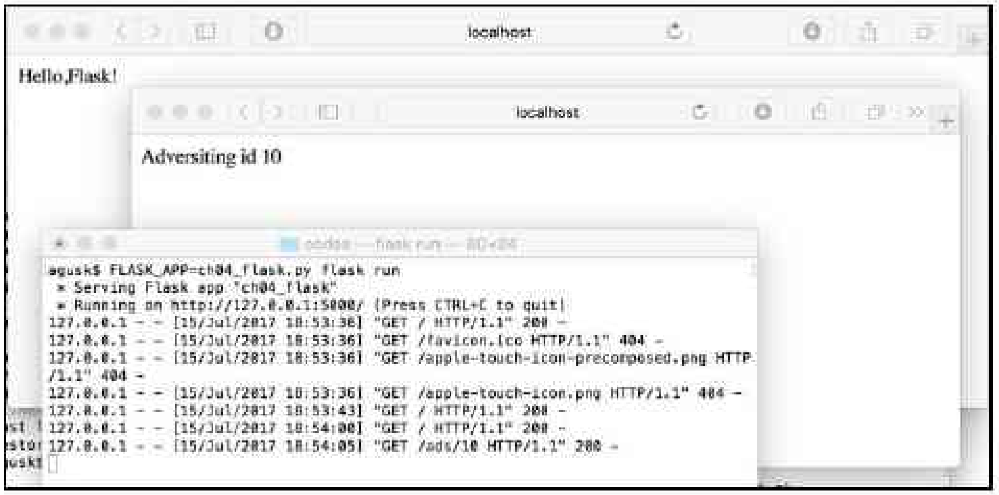
```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello,Flask!"

@app.route('/ads/<int:ads_id>')
def show_post(ads_id):
    return 'Adversiting id %d' % ads_id
```

This program will handle `HTTP GET` requests: `/` and `/ads/<id>`.

To run the program, you can type this command:

```
$ FLASK_APP=ch04_flask.py flask run
```

It shows a running web server on a specific port, by default `5000`. Now you can open a browser and navigate to `http://localhost:5000` and try again to navigate to `http://localhost:5000/ads/10`. You can see the sample browser and program output here:



```
agusk$ FLASK_APP=ch04_flask.py flask run
 * Serving Flask app "ch04_flask"
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [15/Jul/2017 10:53:36] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Jul/2017 10:53:36] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [15/Jul/2017 10:53:36] "GET /apple-touch-icon-precomposed.png HTTP
/1.1" 404 -
127.0.0.1 - - [15/Jul/2017 10:53:36] "GET /apple-touch-icon.png HTTP/1.1" 404 -
127.0.0.1 - - [15/Jul/2017 10:53:43] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Jul/2017 10:54:00] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Jul/2017 10:54:05] "GET /ads/10 HTTP/1.1" 200 -
```

For further information about Flask development, I recommend you read the Flask documentation at http://flask.pocoo.org/docs/. Some program samples are provided to help you get started.

Pyramid

`Pyramid` is a web framework to build web applications based on Python. It's easier to use and to handle HTTP requests. You can install pyramid by typing this command:

```
$ pip install "pyramid==1.9.1"
```

It will install pyramid version 1.9.1. You can change it.

For testing, we'll build a simple web application to handle the `/` request. Use the following code:

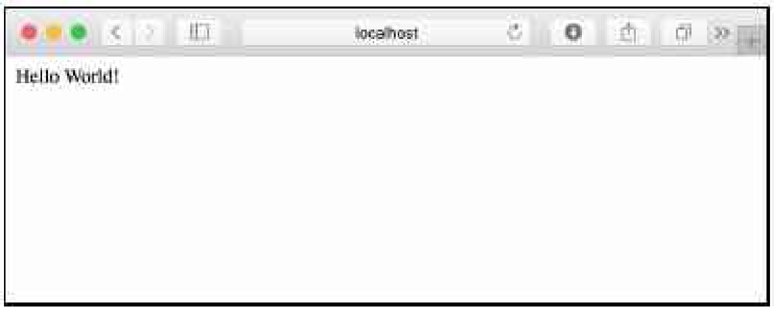
```
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
from pyramid.response import Response

def hello_world(request):
    return Response('Hello World!')

if __name__ == '__main__':
    with Configurator() as config:
        config.add_route('hello', '/')
        config.add_view(hello_world, route_name='hello')
        app = config.make_wsgi_app()

    print('server is running')
    server = make_server('0.0.0.0', 8099, app)
    server.serve_forever()
```

Save this program to a file called `ch04_pyramid.py`.

Now you can run it. Type this command on the terminal.

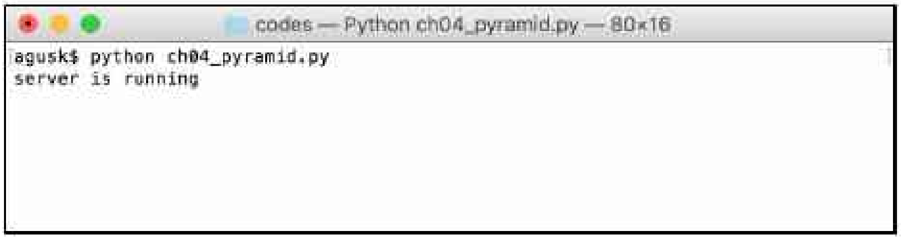
```
$ python ch04_pyramid.py
```

The program will run on port `8099`. Open a browser and navigate to `http://localhost:8099`. You should get a response saying Hello World! in the browser:



```
Hello World!
```

On the terminal side, you can see the running program here:



```
agusk$ python ch04_pyramid.py
server is running
```

Django

Django is a popular web framework for building web applications. You can use it to deliver ad content. Django provides a complete API to build a web application, including database access.

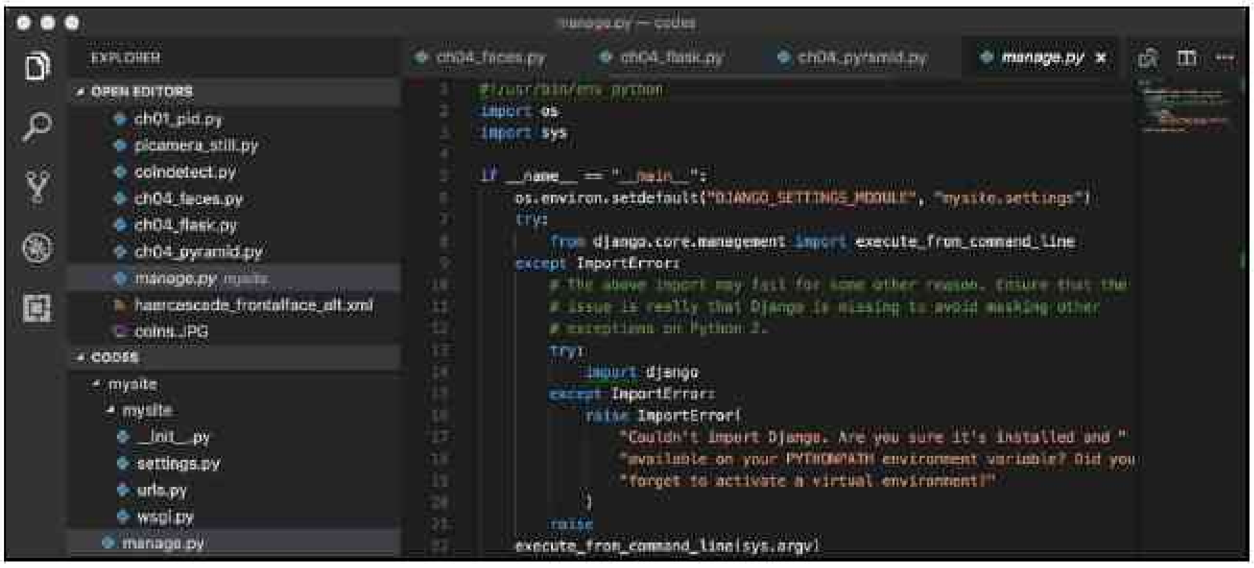
You can install it officially via `pip`. Type this command:

```
$ pip install Django
```

For testing, we'll build a simple web application. We can create a new web project using `django-admin`. For instance, we'll create a project called `mysite`. You can type this command:

```
$ django-admin startproject mysite
```

It will generate a web application with a default template. The program will create the `<project_name>` folder. You can see the project structure here:

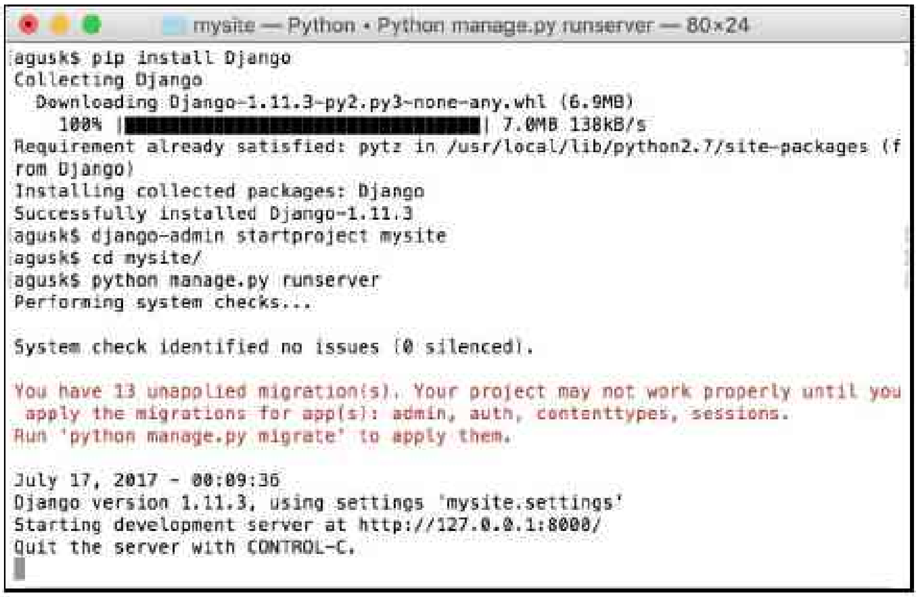


```
#!/usr/bin/env python
import os
import sys

if __name__ == "__main__":
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "mysite.settings")
    try:
        from django.core.management import execute_from_command_line
    except ImportError:
        # The above import may fail for some other reason. Ensure that the
        # issue is really that Django is missing to avoid masking other
        # exceptions on Python 2.
        try:
            import django
        except ImportError:
            raise ImportError(
                "Couldn't import Django. Are you sure it's installed and "
                "available on your PYTHONPATH environment variable? Did you "
                "forget to activate a virtual environment?"
            )
        raise
    execute_from_command_line(sys.argv)
```

We can run this program by typing the following commands:

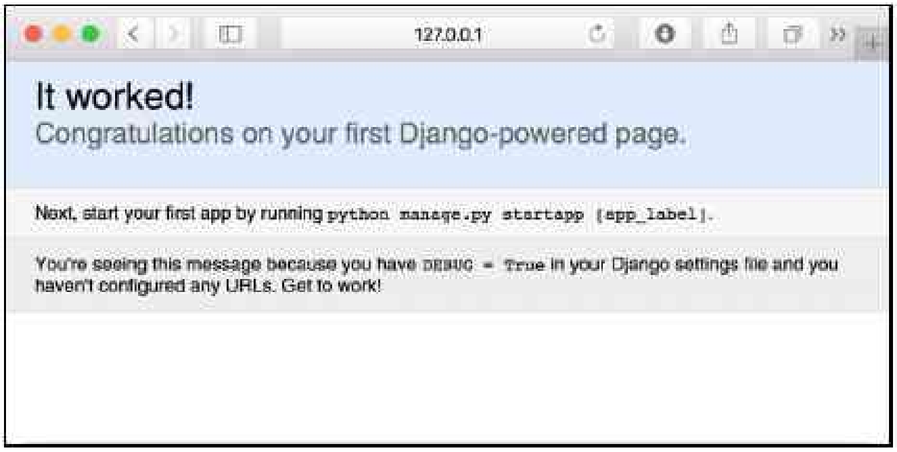
```
$ cd mysite/
    $ python manage.py runserver
```

If you succeeded, you should see the following response message from the program:



```
agusk$ pip install Django
Collecting Django
  Downloading Django-1.11.3-py2.py3-none-any.whl (6.9MB)
    100% |████████████████████████████████| 7.0MB 138kB/s
Requirement already satisfied: pytz in /usr/local/lib/python2.7/site-packages (f
rom Django)
Installing collected packages: Django
Successfully installed Django-1.11.3
agusk$ django-admin startproject mysite
agusk$ cd mysite/
agusk$ python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).

You have 13 unapplied migration(s). Your project may not work properly until you
 apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.

July 17, 2017 - 00:09:36
Django version 1.11.3, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

By default, Django runs on port `8000`. Now you can open a browser and navigate to `http://localhost:8000`. If successful, you should see a simple web application:



It worked!
Congratulations on your first Django-powered page.

Next, start your first app by running `python manage.py startapp [app_label]`.

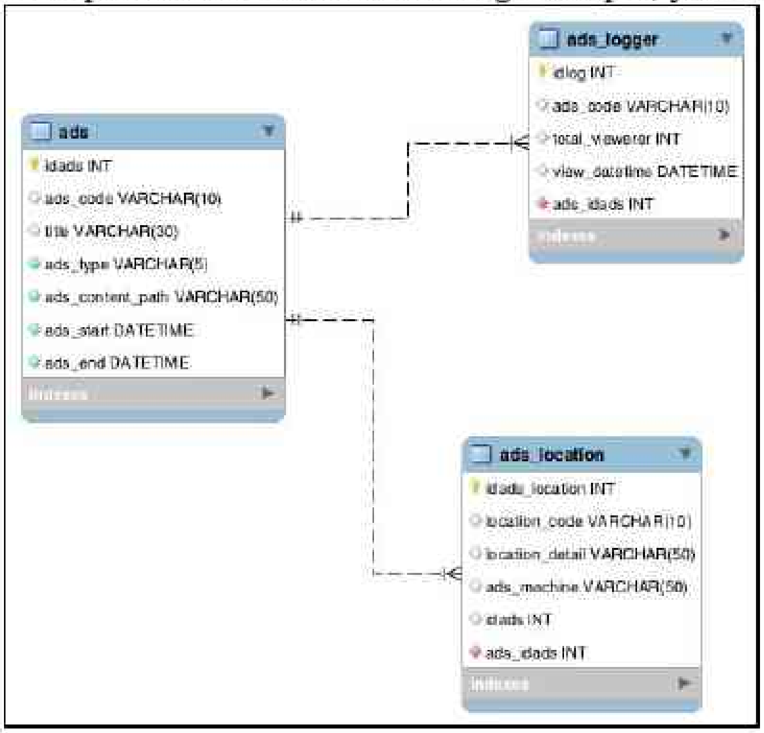
You're seeing this message because you have `DEBUG = True` in your Django settings file and you haven't configured any URLs. Get to work!

Building a smart digital advertising dashboard

Basically, building a smart digital advertising dashboard is the same as building a digital signage system. We use existing signage system platforms that fit your case. Depending on your requirements, we can implement digital advertising using Raspberry Pi with HDMI monitor.

To make our digital signage system smart, we need added values. One of them is to add a sensor to detect human presence that we learned on previous section.

With respect to data, we should design a database model that covers our sensor data. For instance, we can add information about the number of viewers in a certain time period. For a database design sample, you can see the following figure:



You can see from this database design that the number of viewers is represented as `total_viewer` in the `ads_logger` table. Each ad content should be defined by how and when the content will be shown. In the `ads` table, we set the `ads_type` for the ad content type. It can be text, image, or video. We also define `ads_start` and `ads_end` in the `ads` table to define the ad display period.

This is a sample database design. You can extend it with additional features, such as a billing system. Consider you can implement charging model for companies that want to apply ads. This charging model can use showing time and location.

Summary

In this chapter, we learned how to create a digital signage system with Artificial Intelligence with implementing human presence. Various human presence methods were also explored to provide added value to our digital signage system.

In the next chapter, we will explore a smart speaker machine and the technology that is used to build such a system.

A Smart Speaker Machine

Nowadays, speech and voice technology is applied to various IoT platforms. We can see Amazon and Google already having built smart speaker machines, Amazon Echo and Google Home. In this chapter, we will explore and learn how to build a smart speaker machine. For development and testing, we will use the ReSpeaker board to implement a smart speaker machine.

In this chapter, we'll learn the following topics:

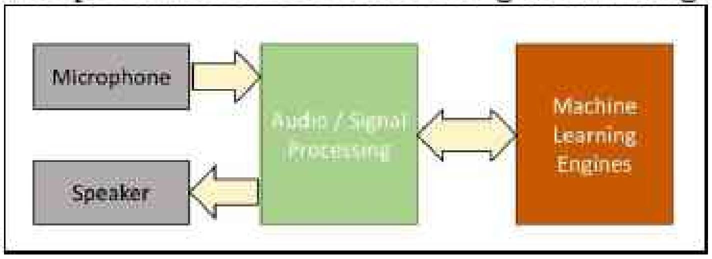
- Introducing smart speaker machines
- Exploring existing smart speaker machines
- Introducing ReSpeaker
- Integrating your IoT boards with ReSpeaker
- GPIO programming on ReSpeaker
- Connecting to the Microsoft Bing Speech API
- Building your own smart speaker machine

Let's explore!

Introducing smart speaker machines

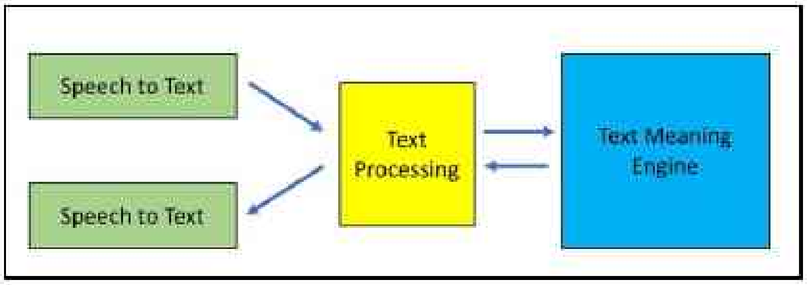
In recent years, big companies such as Amazon, Apple, Google, and Microsoft have invested in speech technology. Speech technology implementation can be through software, hardware, and hybrid software and hardware. Speech technology usually uses Artificial Intelligence methods to detect and recognize speech or voice and then perform something based on the speech/voice input. Amazon Echo and Google Home are samples of speech technology implementations in hybrid hardware and software. They can be called smart speaker machines.

In general, a smart speaker machine consists of microphone and speaker devices as sensor and actuator. The speaker can record human voice and then convert it to analog values. A speaker can be used to generate sounds based on signal parameters such as frequency and amplitude. Human voice in analog form is converted to digital form so we can process it easily. In digital form, we can implement various algorithms to it in a computer. One of these tasks is to convert human speech to text. It is usually called speech-to-text. Alternatively, we also can synthesize human voice from text. We need a specific algorithm based in AI to convert text to human speech/voice. You can see a general design of a smart speaker machine here:



Designing a smart speaker machine usually involved integrating a machine learning program such as speech-to-text and text-to-speech. In this case, we'll build a program to recognize human speech in digital form and then convert it into text.

After we obtain text from human speech, we can perform text processing. For instance, if we get the text "turn on LED", we'll perform actions to turn on the LED. It involves text processing to obtain meaning from text. You can see a general design of a text meaning system here:


Speech to Text
Speech to Text
Text Processing
Text Meaning Engine

Exploring existing smart speaker machines

Manufacturers make smart speaker machines to perform automation tasks. In this section, we will explore various smart speaker machines.

Amazon Echo

Amazon Echo is a smart hardware system. Amazon has used an artificial intelligence engine in Amazon Echo to enable interaction with users. This program is called **Alexa**. We can perform tasks on Amazon Echo through voice commands. There are some keywords that are used by Alexa to identify and recognize voice commands. Currently, we can buy Amazon Echo from https://www.amazon.com/dp/B00X4WHP5E/. You can see Amazon Echo here:

Amazon also provides a cheap hardware device for Amazon Echo, called Amazon Echo Dot (https://www.amazon.com/dp/B01DFKC2SO/). The model's size is small and it is 163 grams in weight. You can see Amazon Echo here:

To start communicating with Amazon Echo, we should say *Alexa*. This is a keyword using which Amazon Echo starts to listen to commands. After recording the commands, Amazon Echo will interpret these commands to perform something.

Google Home

Google Home is a smart speaker machine powered by Google Assistant. This device can guide you based on your experience. As we know, Google has a lot of information from heterogeneous sources, so we can gather information from Google Home by giving it voice commands, including home automation tasks.

Like Amazon Echo, Google Home needs specific voice commands to perform any task. Some languages are supported, and English is the default language. For further information about Google Home, you can visit the official website at https://madeby.google.com/home/. You can see Google Home here:

Ivee

Ivee is a personal voice assistant. Like other machines, Ivee applies **natural language speech** (**NLP**) processing with their proprietary algorithms to recognize and understand our speech. Ivee also has an advantage in size. If you're interested, you can visit the official website at https://helloivee.com. Here is what the Ivee looks like:

Triby

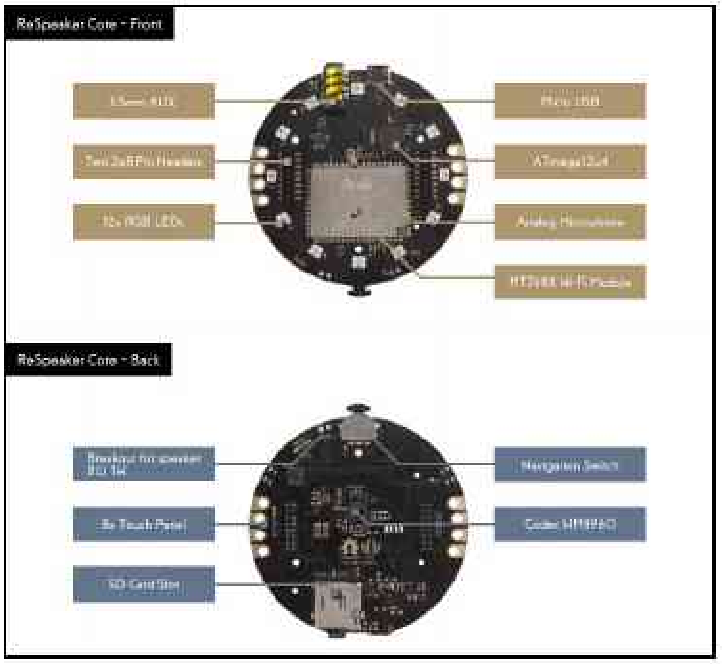
Triby is a smart speaker machine that has a built-in Alexa voice service, so the machine can recognize and understand our speech. Since Triby machine use the same services as Amazon Echo, we can get more benefits. In addition, Triby applies additional features and services, such as an e-paper screen so you can read messages or display something. Interested? You can visit the official website at http://www.invoxia.com/triby/. Here's a picture of the Triby:

Introducing ReSpeaker

Technically, making a smart speaker machine is easy. We need a microphone and speaker for input and output audio. We also need an audio processing module and machine learning engine to manipulate speech audio and interpret speech commands.

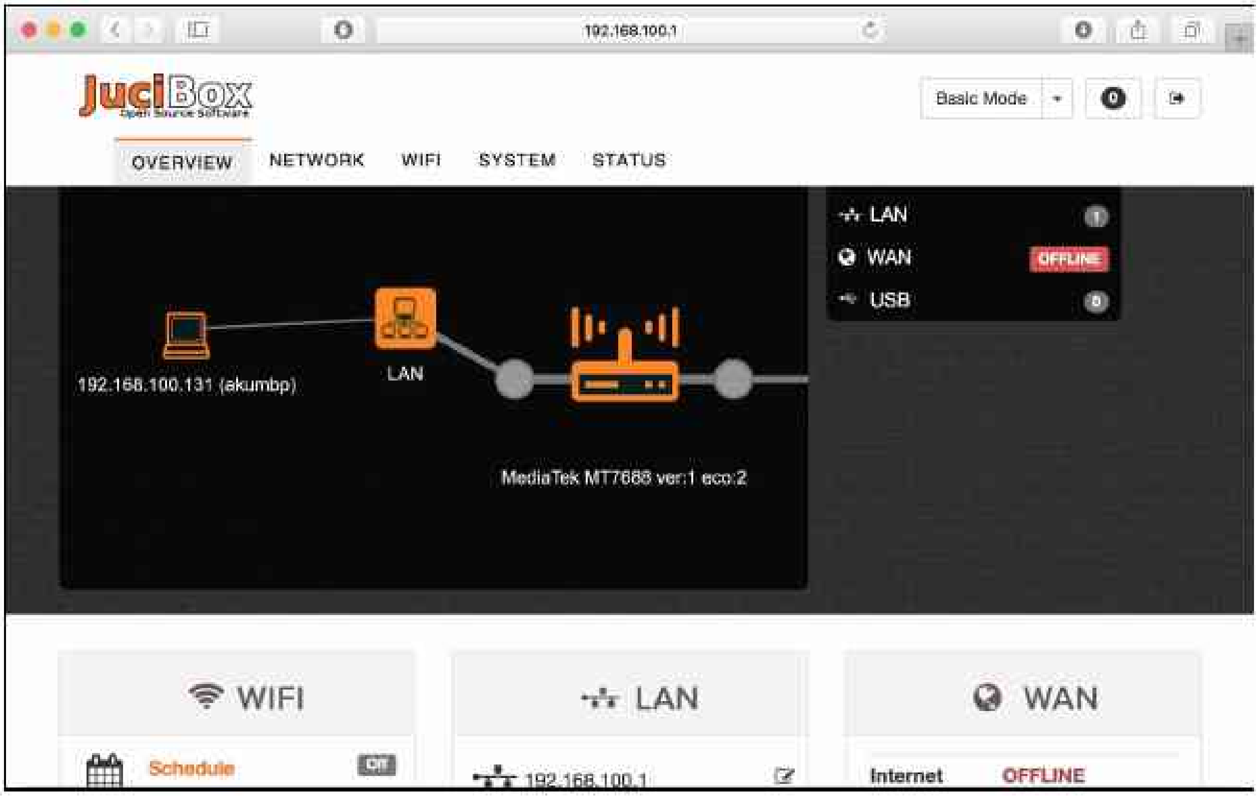
In this section, we'll learn about one of the platforms for a smart speaker machine--ReSpeaker from SeeedStudio. We can use it with IoT boards such as Arduino and Raspberry Pi to perform automation tasks. You can get this module from SeeedStudio at https://www.seeedstudio.com/ReSpeaker-Core-Based-On-MT7688-and-OpenWRT-p-2716.html. Here it is:

The ReSpeaker board uses the AI7688 Wi-Fi module, running the OpenWrt OS. To implement additional automation tasks, ReSpeaker uses the ATMega32U4 as its coprocessor and WM8960 for its codec engine. You can see the ReSpeaker core board layout in the following figure:



Once we connect ReSpeaker to a power adapter or a computer through a micro-USB cable, we can see the Wi-Fi access point from ReSpeaker. It usually shows up as ReSpeakerxxxxxx, where xxxxxx is a random number. Try to connect to this Wi-Fi from your computer. Then, you we will be asked to join ReSpeaker to an existing Wi-Fi network. You can select your existing Wi-FI or even ignore it.

By default, ReSpeaker has the IP address `192.168.100.1` if you connect to the ReSpeaker Wi-Fi. If your ReSpeaker has joined an existing Wi-Fi, you should verify the ReSpeaker IP address. Now you can open a browser and navigate to the IP address of ReSpeaker so you can see the ReSpeaker dashboard, shown in the following screenshot. It shows all states of the ReSpeaker board.



Now we test our ReSpeaker board to build a simple smart speaker machine. If you look at the ReSpeaker core layout, it has a built-in analog microphone. To develop a program in ReSpeaker, the system has provided a Python library to interact with the board. For testing, we'll build a program for speech-to-text using Python.

Open your favorite text editor to write the following Python program:

```
import logging
import time
from threading import Thread, Event

from respeaker import Microphone


def task(quit_event):
    mic = Microphone(quit_event=quit_event)

    while not quit_event.is_set():
        if mic.wakeup('respeaker'):
            print('Wake up')
            data = mic.listen()
            text = mic.recognize(data)
            if text:
                print('Recognized %s' % text)


def main():
    print('ReSpeaker is running..')
    logging.basicConfig(level=logging.DEBUG)
    quit_event = Event()
    thread = Thread(target=task, args=(quit_event,))
    thread.start()
    while True:
        try:
            time.sleep(1)
        except KeyboardInterrupt:
            print('Quit')
            quit_event.set()
            break
    thread.join()

if __name__ == '__main__':
    main()
```

Save this program as `ch05_respeaker_demo.py`.

Transfer this file to ReSpeaker via SFTP. I recommend you use the Filezilla client app. You can download this application from https://filezilla-project.org. After you have uploaded the file to the ReSpeaker board, you can execute this program. You can type this command in the ReSpeaker terminal:

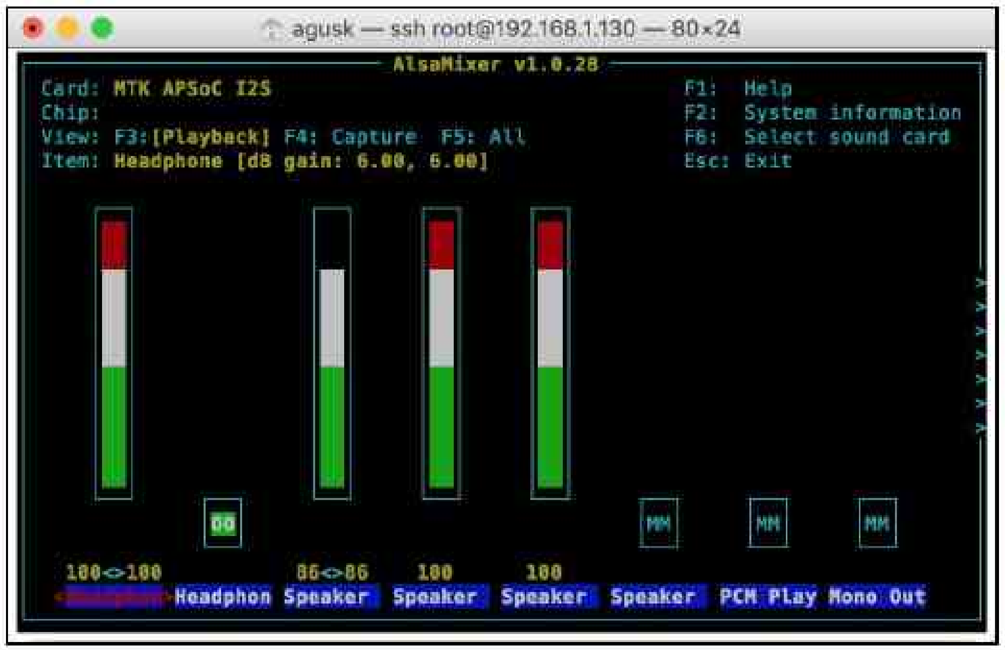
```
$ python ch05_respeaker_demo.py
```

Now you can start to give a command to ReSpeaker by saying *respeaker*. After this, you can say anything and the program will convert it to recognize the speech.

If you get a problem related to the audio channel while running the program, you can configure it using `alsamixer`. Run this on the ReSpeaker terminal:

```
$ alsamixer
```

After its has executed, you should see the AlsaMixer application. Configure the sound card by pressing *F6*. Once done, you can press the *Esc* key to exit the program. The AlsaMixer application is shown here:



```
agusk — ssh root@192.168.1.130 — 80×24
──────────────────────── AlsaMixer v1.0.28 ────────────────────────
Card: MTK APSoC I2S                              F1:  Help
Chip:                                            F2:  System information
View: F3:[Playback] F4: Capture  F5: All         F6:  Select sound card
Item: Headphone [dB gain: 6.00, 6.00]            Esc: Exit

                                  OO                                    MM      MM      MM

 100<>100              86<>86   100      100
<Headphon>Headphon Speaker  Speaker  Speaker  Speaker  PCM Play Mono Out
```

```
mic = Microphone(quit_event=quit_event) while not
quit_event.is_set(): if mic.wakeup('respeaker'): print('Wake up')

data = mic.listen() <br/>text = mic.recognize(data)<br/>if text:<br/>
print('Recognized %s' % text)
```

Integrating your IoT boards with ReSpeaker

ReSpeaker is built with Arduino (ATmega32U4) and Linux-based OpenWrt with MCU MT7688 so that we can access GPIO pins in our program. We can develop a sketch program for the ReSpeaker board. Start by downloading the ReSpeaker library for Arduino from https://github.com/respeaker/respeaker_arduino_library. Download and extract it to the Arduino library with the name respeaker.

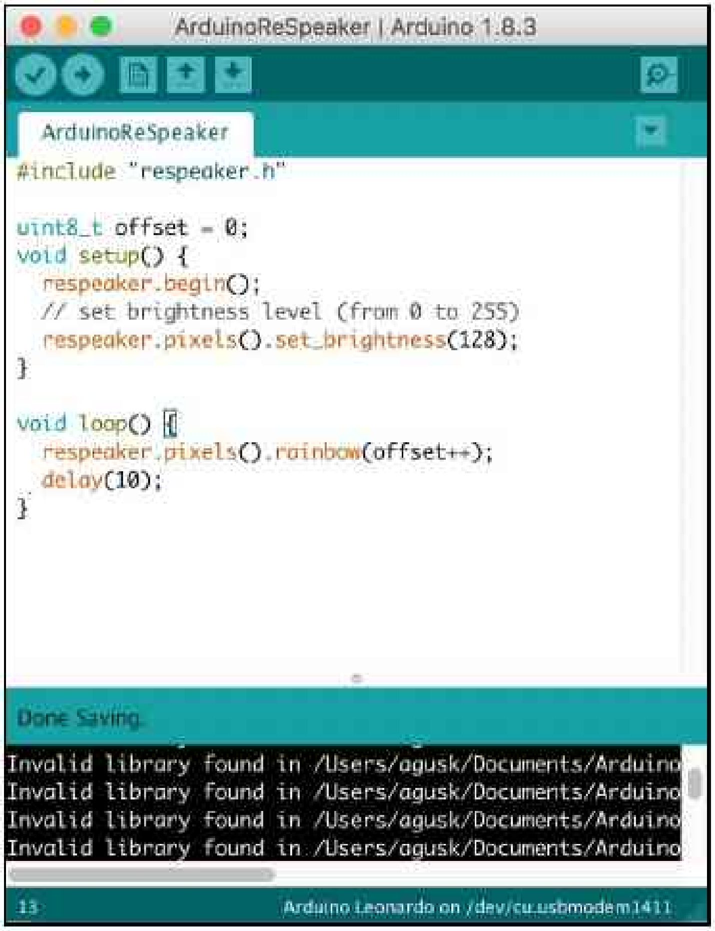
Now you can use the ReSpeaker library in your Arduino IDE. For testing, we'll try to access 12 RGB LEDs using the pixels library.

First, you should install Arduino software from this site: https://www.arduino.cc/en/Main/Software. Then, you can write the following sketch program:

```
#include "respeaker.h"

uint8_t offset = 0;
void setup() {
  respeaker.begin();
  // set brightness level (from 0 to 255)
  respeaker.pixels().set_brightness(128);
}

void loop() {
  respeaker.pixels().rainbow(offset++);
  delay(10);
}
```

This program starts to initialize the ReSpeaker library by calling `begin()` from the `respeaker` object. In the `loop()` function, we change colors of the RGB LEDs.

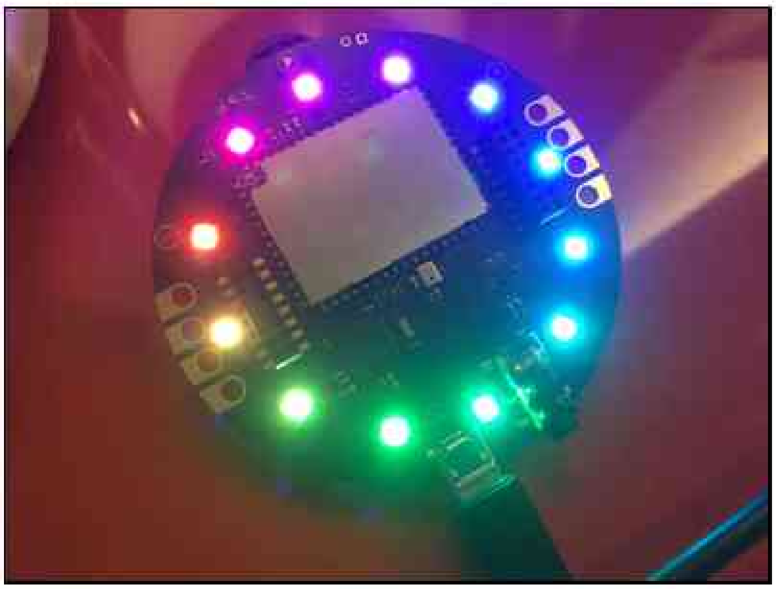
Save this sketch as `ArduinoReSpeaker`. In order to upload the sketch program to ReSpeaker, you should configure the target board to Arduino Leonardo and change the port to your ReSpeaker port.



```
#include "respeaker.h"

uint8_t offset = 0;
void setup() {
  respeaker.begin();
  // set brightness level (from 0 to 255)
  respeaker.pixels().set_brightness(128);
}

void loop() {
  respeaker.pixels().rainbow(offset++);
  delay(10);
}
```

```
Done Saving.
Invalid library found in /Users/agusk/Documents/Arduino
Invalid library found in /Users/agusk/Documents/Arduino
Invalid library found in /Users/agusk/Documents/Arduino
Invalid library found in /Users/agusk/Documents/Arduino
```

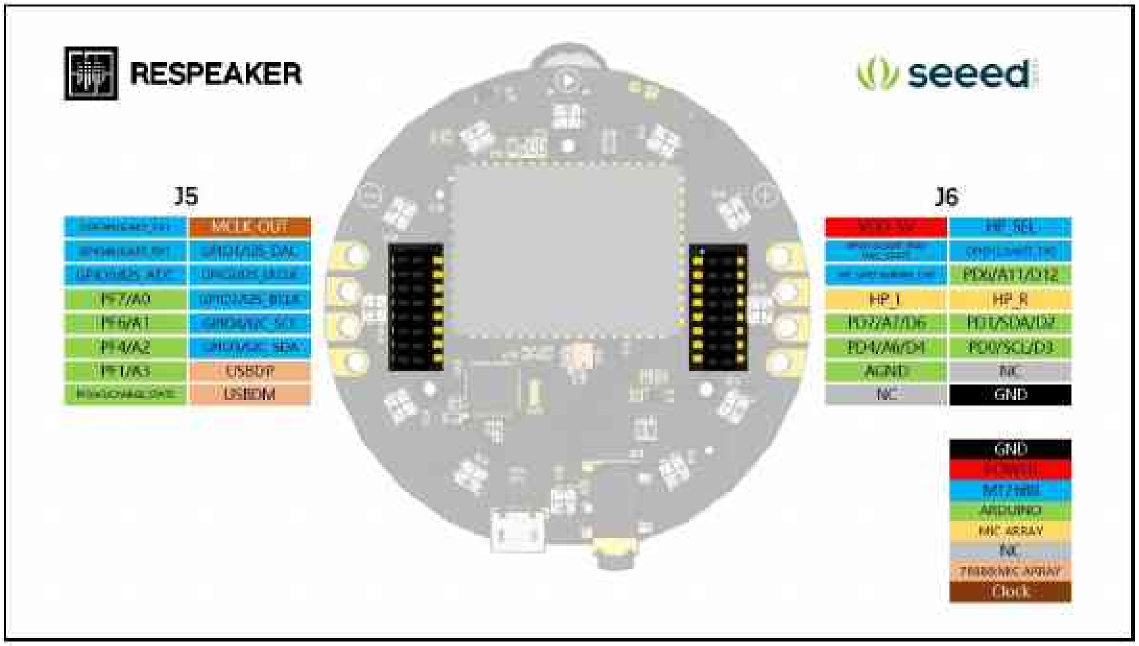
After uploading the program, you should see all RGB LEDs changing their color gradually. You can see it here:



GPIO programming on ReSpeaker

ReSpeaker has one MCU MT7688 and co MCU ATmega32U4, so we can access both MCUs from our program. Inside the board, ReSpeaker runs Linux OpenWrt, so we can perform Linux operations on the ReSpeaker terminal. Not all GPIO pins are accessible from a program. ReSpeaker exposes specific GPIOs.

In general, we can use the following GPIO layout of the ReSpeaker board:

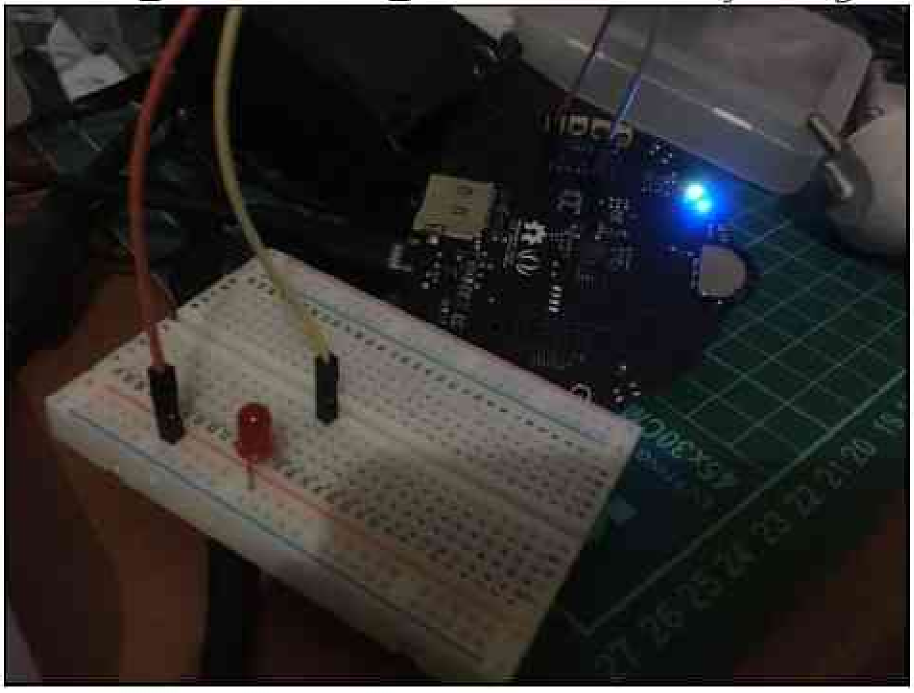


You can see here that some GPIO pins belong to MCU MT7688 and MCU ATmega32U4. If you want to know the complete schematic of the ReSpeaker core board, I recommend you read this document at https://github.com/respeaker/get_started_with_respeaker/blob/master/Introduction.md#hardware.

To access GPIO pins on MCU ATmega32U4, you can use Arduino software. We can write a sketch for Arduino program in the MCU ATmega32U4.

For MCU MT7688, we can access GPIO using GPIO programming for Linux since ReSpeaker uses Linux OpenWrt.

For testing, we'll connect an LED to GPIO on MCU MT7688. You can connect it on MT_GPIO18/PWM_CH0. You can see my wiring here:



Let's start writing a program using GPIO with the Linux approach. Open the ReSpeaker terminal. Since we use GPIO18 on MCU MT7688, we activate it with the output direction. Type these commands: **$ echo 18 > /sys/class/gpio/export $ echo "out" > /sys/class/gpio/gpio18/direction**

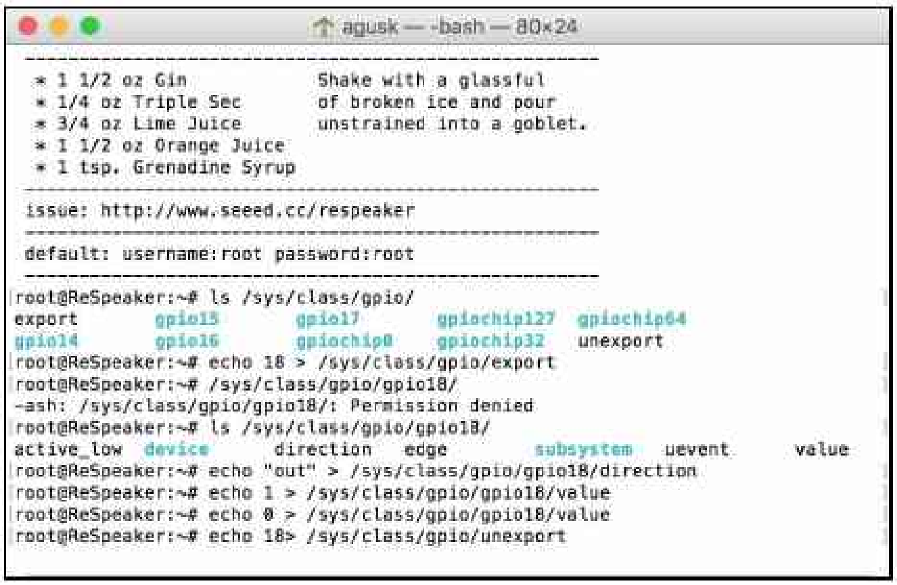
In this case, GPIO18 is activated for output mode.

Now you can set values for HIGH and LOW as `1` and `0`. You can type these commands on the ReSpeaker terminal: **$ echo 1 > /sys/class/gpio/gpio18/value $ echo 0 > /sys/class/gpio/gpio18/value**

You should see the LEDs light up.

If you don't need this GPIO anymore, you can release it so other programs can access it. Type this command on the ReSpeaker terminal: **$ echo 18> /sys/class/gpio/unexport**

You can see my program output on the ReSpeaker terminal here:



```
 ----------------------------------------------------
  * 1 1/2 oz Gin           Shake with a glassful
  * 1/4 oz Triple Sec      of broken ice and pour
  * 3/4 oz Lime Juice      unstrained into a goblet.
  * 1 1/2 oz Orange Juice
  * 1 tsp. Grenadine Syrup
 ----------------------------------------------------
 issue: http://www.seeed.cc/respeaker
 ----------------------------------------------------
 default: username:root password:root
 ----------------------------------------------------
root@ReSpeaker:~# ls /sys/class/gpio/
export       gpio15       gpio17       gpiochip127  gpiochip64
gpio14       gpio16       gpiochip0    gpiochip32   unexport
root@ReSpeaker:~# echo 18 > /sys/class/gpio/export
root@ReSpeaker:~# /sys/class/gpio/gpio18/
-ash: /sys/class/gpio/gpio18/: Permission denied
root@ReSpeaker:~# ls /sys/class/gpio/gpio18/
active_low  device      direction   edge        subsystem   uevent      value
root@ReSpeaker:~# echo "out" > /sys/class/gpio/gpio18/direction
root@ReSpeaker:~# echo 1 > /sys/class/gpio/gpio18/value
root@ReSpeaker:~# echo 0 > /sys/class/gpio/gpio18/value
root@ReSpeaker:~# echo 18> /sys/class/gpio/unexport
```

We also can develop a Python program to access GPIO on MT7688. In general, ReSpeaker already provides an installed Python library for MT7688. You read about it at https://github.com/respeaker/respeaker_python_library.

Extending our previous demo, we'll try to turn on/off LEDs on GPIO18 using Python. Type this script:

```
from respeaker import gpio

gpio18 = gpio.Gpio(18, gpio.DIR_OUT)
# turn on LED
gpio18.write(1)
# turn off LED
gpio18.write(0)

# close gpio
gpio18.close()
```

Save it as `ch05_respeaker_gpio.py`. You can run this program by typing this command on the ReSpeaker terminal:

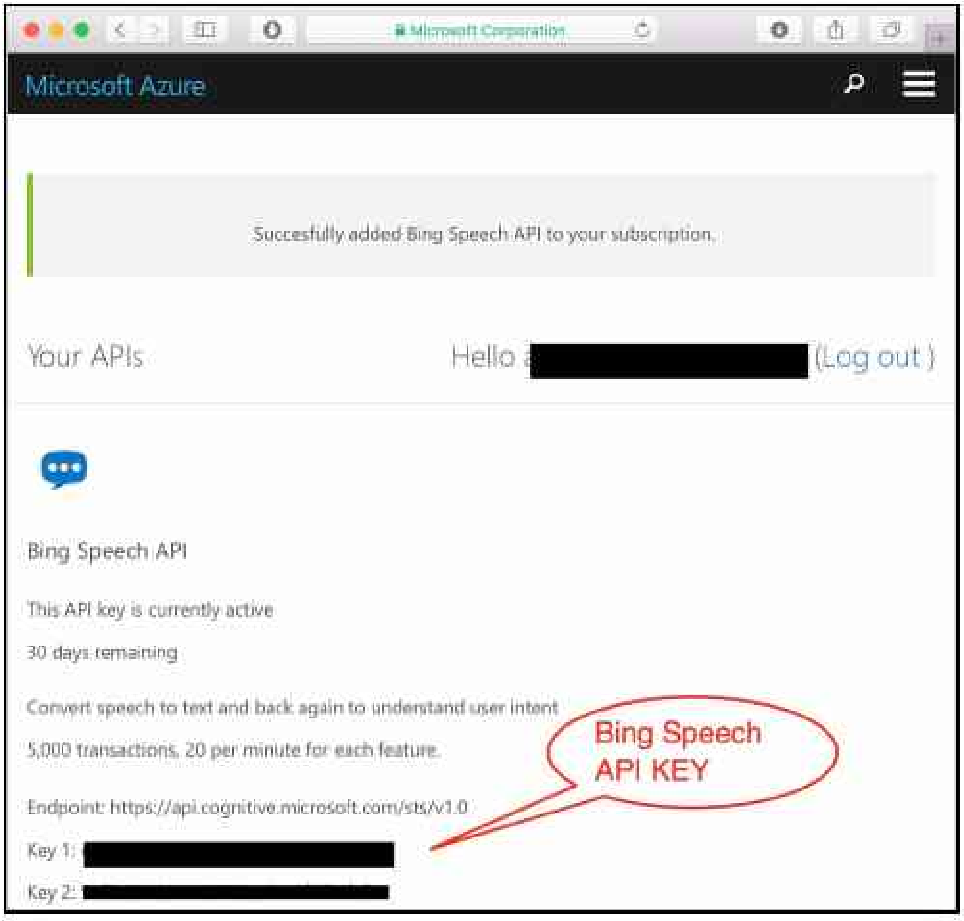
```
$ python ch05_respeaker_gpio.py
```

Now you can see LEDs turning on and off.

Connecting to the Microsoft Bing Speech API

ReSpeaker provides connectivity with the Microsoft Bing Speech API. Using this library, we can apply speech recognition such as speech-to-text. For further information about the Microsoft Bing Speech API, you can visit at https://azure.microsoft.com/en-us/services/cognitive-services/speech/.

To use the Microsoft Bing Speech API library, you should register and obtain an API key. Microsoft provides trial access to use. The API key can be found on your dashboard page of the Microsoft Bing Speech API. You can see it here:



Succesfully added Bing Speech API to your subscription.

Your APIs

Hello [redacted] (Log out)

Bing Speech API

This API key is currently active

30 days remaining

Convert speech to text and back again to understand user intent

5,000 transactions, 20 per minute for each feature.

Endpoint: https://api.cognitive.microsoft.com/sts/v1.0

Key 1: [redacted]

Key 2: [redacted]

Now we can write a Python program to use the Microsoft Bing Speech API:

```
import logging
import time
from threading import Thread, Event
from respeaker import Microphone
from respeaker.bing_speech_api import BingSpeechAPI

# get a key from https://www.microsoft.com/cognitive-services/en-us/speech-api
BING_KEY = '<--bing speech api-->'

def task(quit_event):
    mic = Microphone(quit_event=quit_event)
    bing = BingSpeechAPI(key=BING_KEY)

    while not quit_event.is_set():
        if mic.wakeup('respeaker'):
            print('Wake up')
            data = mic.listen()
            try:
                text = bing.recognize(data)
                if text:
                    print('Recognized %s' % text)
            except Exception as e:
                print(e.message)

def main():
    print('ReSpeaker is running....')
    logging.basicConfig(level=logging.DEBUG)
    quit_event = Event()
    thread = Thread(target=task, args=(quit_event,))
    thread.start()
    while True:
        try:
            time.sleep(1)
        except KeyboardInterrupt:
            print('Quit')
            quit_event.set()
            break
    thread.join()

if __name__ == '__main__':
    main()
```

Save this program into a file called `ch05_respeaker.py`.

To run this program, you can type this command:

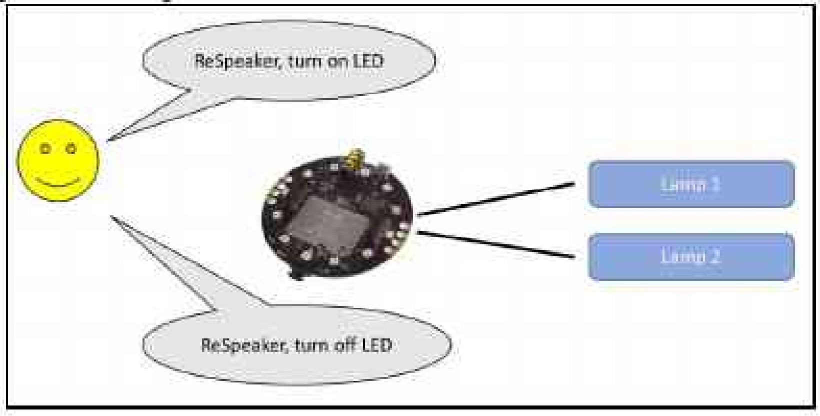
```
$ python ch05_respeaker.py
```

Since this program uses the Microsoft Bing Speech API, your ReSpeaker should be connected to the internet.

After running it, say *respeaker* until the program wakes up. Then, say something so the program converts speech to text. Speak slowly to make sure ReSpeaker can recognize your voice.

Building your own smart speaker machine

We've already learned how to work with ReSpeaker and develop programs for it. We also can develop a smart speaker machine. You can see our scenario in the following diagram:



You can see that the ReSpeaker core board connects to lamps. You can see simple a LED (DC) or lamp (AC) with a relay module.

If you use the Microsoft Bing Speech API, you can develop a smart speaker machine for recognizing some phrases such as *turn on* and *turn off*. After you obtain the text, you can parse it to determine whether it contains *turn on* or *turn off*. This is a basic smart speaker machine. You can customize it based on your use case. For instance, you build an automation system with Arduino/Raspberry Pi. Then, ReSpeaker will be used as speech input. If person says specific word, ReSpeaker will send command to Arduino/Raspberry Pi to perform something.

Summary

We learned about smart speaker machines using the ReSpeaker core board. Some samples are provided to get started with ReSpeaker core.

In the next chapter, we will explore an autonomous firefighter robot and the technology used to build it.

Autonomous Firefighter Robot

Robot development is one of the big challenges in science and technology. It involves multidisciplinary fields of science and technology. In this chapter, we'll learn and explore how to build an autonomous robot for firefighting, starting with exploring robot platforms and then extending robot capabilities to address firefighting.

We'll learn the following topics:

- Introducing autonomous firefighter robots
- Exploring robot platforms
- Detecting a fire source
- Basic remote robot navigation
- Detecting obstacles
- Designing an autonomous robot
- Building an autonomous firefighter robot

Let's explore!

Introducing autonomous firefighter robots

An autonomous robot is one that has its own decision-making capacity to perform tasks such as moving, stopping, pressing a button, or some other specific purpose. Autonomous robots consist of artificial intelligence programs to perform decision computing. Researchers are interested in building a new model for autonomous robots. The more general we make an autonomous robot model, the more attention and effort goes in its development. However, autonomous robot development still needs a specific purpose so it can execute its goals.

An autonomous firefighter robot is one autonomous robot model with the specific purpose of finding and putting out fires. In this chapter, we'll learn and explore how to build an autonomous robot for firefighting.

A sample firefighter robot implementation is Thermite 3.0. This robot has a water sprinkler to extinguish fires. You can explore this robot on the official website, http://www.firefightrobot.com. This is what it looks like:

Exploring robot platforms

In this section, we'll explore various existing robot platforms that can be applied to build autonomous robots. We will review robot platforms with Arduino and Raspberry capabilities as the robot computing core.

Zumo robot for Arduino

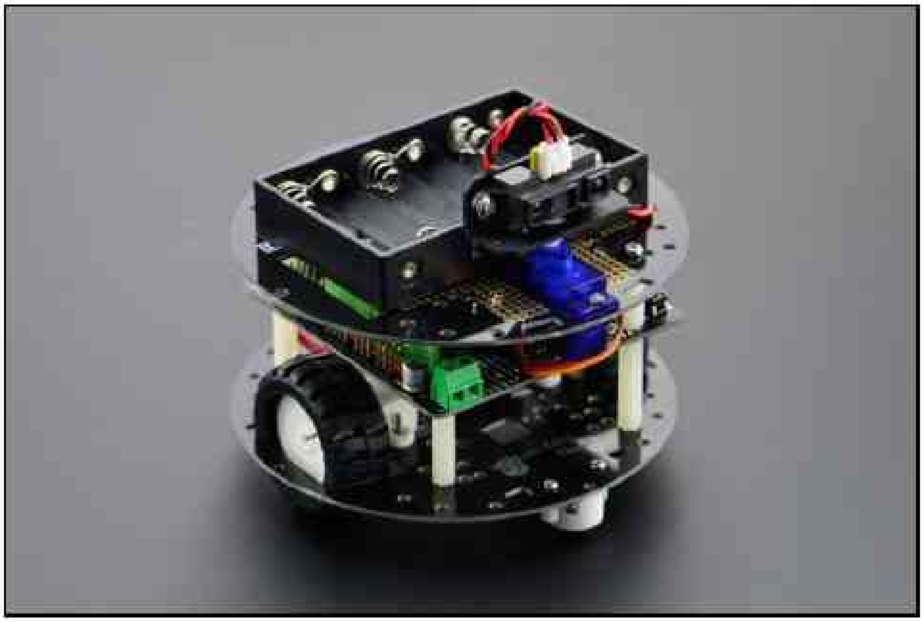
Pololu are an electronics manufacturer and online retailer. They provide various robot platforms. You can see a list of robot platforms at https://www.pololu.com/category/2/robot-kits. Some robot platforms are available as robot kits without soldering. This means you can get your robot up and running without performing soldering on the kit. You can see a list of solder-less robot kits at https://www.pololu.com/category/4/robot-kits-without-soldering.

One robot kit you can make without soldering is the Zumo robot for Arduino. This kit enables your Arduino-based Arduino UNO model to be attached to the board. If you are interested in this robot kit, you can check it out on https://www.pololu.com/product/2510. The Zumo robot for Arduino can be seen here:

MiniQ Discovery Arduino robot kit

The MiniQ Discovery Arduino robot kit is a robotic platform that is ready to use. This kit is manufactured by DFRobot. They have made custom a Arduino board, called the Romeo board, to manage the robot. This board has a built-in motor driver to manage motor movement.

If you are interested in this kit, you can visit https://www.dfrobot.com/product-1144.html. You can see the MiniQ Discovery Arduino robot kit the following image. In this book, I will use this robot kit for my implementation.



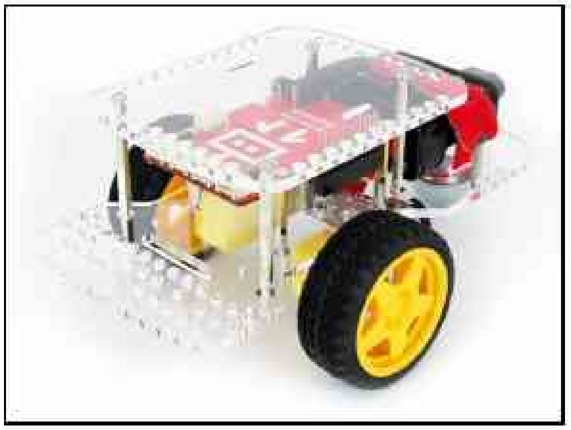
Turtle kit - a 2WD DIY Arduino robotics kit for beginners

Another robot kit from DFRobot is Turtle kit. This robot uses the Romeo BLE board, which consists of an Arduino UNO model with a BLE module. The Romeo BLE board can be controlled from a smartphone through a BLE network.

You can see the Turtle kit in the following figure. Its website is https://www.dfrobot.com/product-1225.html.

GoPiGo robot

GoPiGo is a robot platform-based Raspberry Pi that is manufactured by Dexter industries. They provide a basic kit that you can bring your own Raspberry Pi to. It's called the GoPiGo robot Base Kit: https://www.dexterindustries.com/shop/gopigo3-robot-base-kit/. You can see its form here:

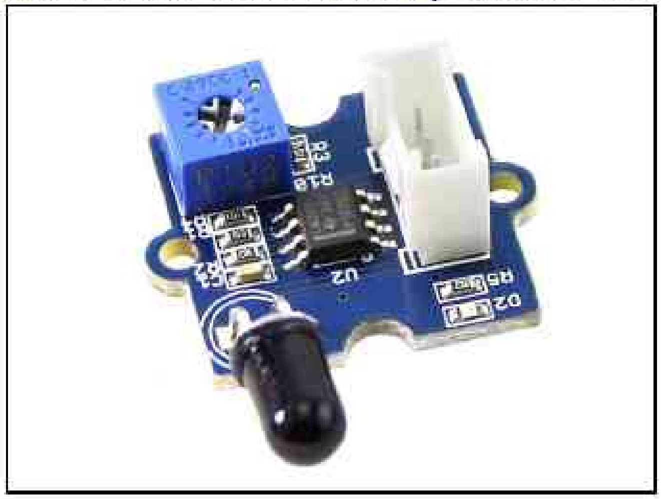


Detecting a fire source

One of requirements of building an autonomous firefighter robot is to find a fire's source. Fire-detection sensors are available for Arduino and Raspberry Pi. In this section, we'll explore fire-detection sensors.

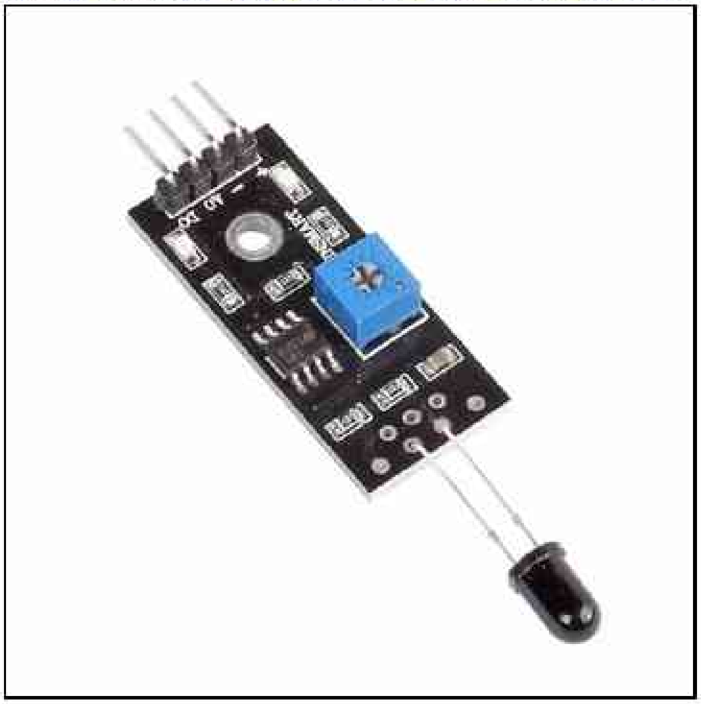
Grove - flame sensor

If you have a Grove shield from SeeedStudio, we can use the Grove flame sensor to detect a fire. Technically, this sensor uses the YG1006 sensor with high photosensitivity. This sensor is easy to use on Arduino and Raspberry Pi. If you are interested, you can visit the website to obtain further information: https://www.seeedstudio.com/Grove-Flame-Sensor-p-1450.html. You can see the Grove flame sensor here:



SainSmart flame detection sensor module

If you don't have a Grove shield for a flame sensor, you can use any flame sensor module, for instance, the flame sensor module from SainSmart. This sensor module enables the Arduino and Raspberry Pi to be attached to the board. This sensor module is cheap, and you can find it at https://www.sainsmart.com/ir-infrared-flame-detection-sensor-module-detect-fire-flame-sensor-for-arduino.html. This is how it looks:



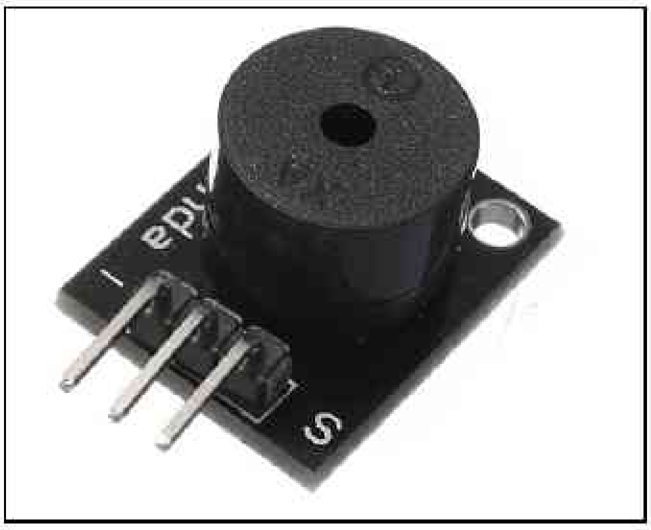
Basically, a flame sensor is cheap and you can find it on Aliexpress, Banggood, or another online store. A flame sensor usually has three or four pins: GND, VCC, A0, and D0. Some flame sensors have the A0 pin or D0 pin or both.

Demo - fire detection

In this section, we'll develop a simple application to detect fire. In this case, we need two modules: flame sensor and buzzer. We will implement this demo using an Arduino board.

Our scenario is to read a fire-detection value from the flame sensor. If the sensor reading reaches a certain value, Arduino will turn on a buzzer device to generate a sound.

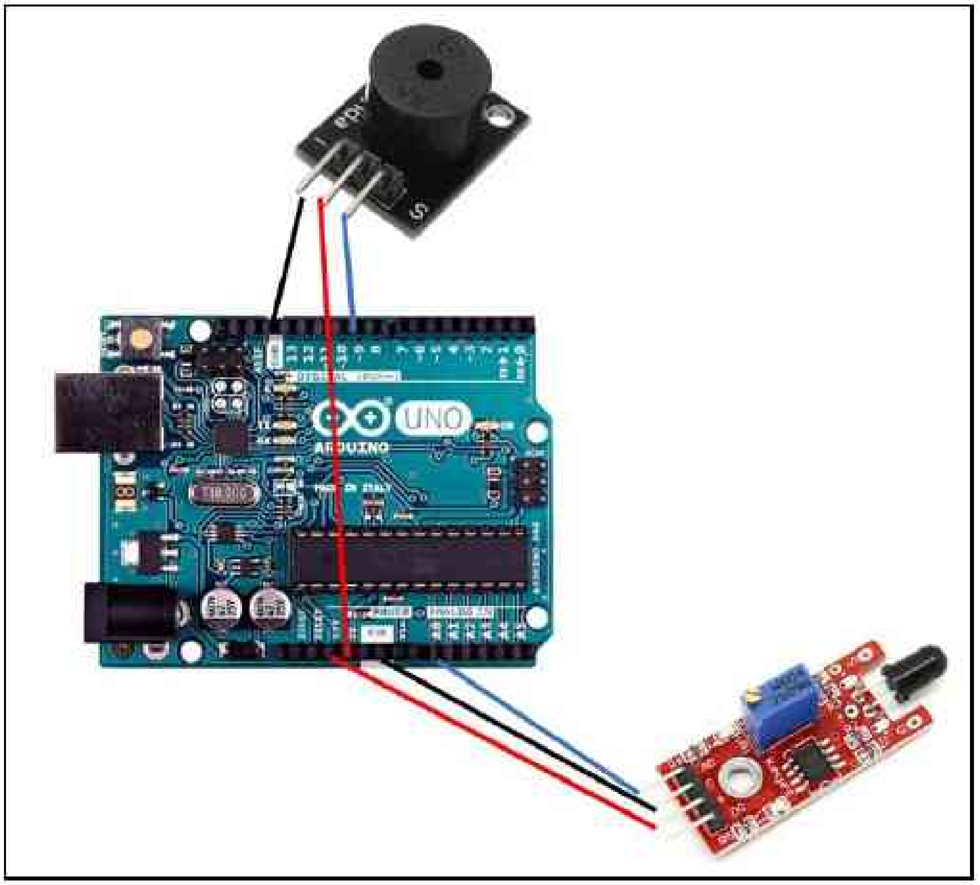
In general, the buzzer device has three pins: GND, VCC and Signal. We can connect the buzzer device to PWM pins on the Arduino. You can see the buzzer device here:



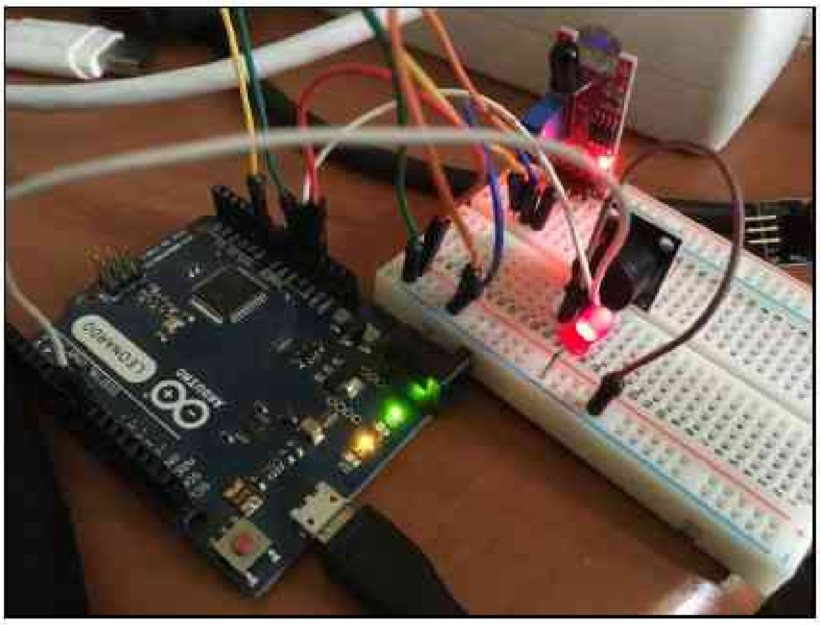
Let's start to implement the hardware wiring. Use the following connections:

- Buzzer GND pin connected to Arduino GND pin
- Buzzer VCC pin connected to Arduino 5V pin
- Buzzer S (signal) pin connected to Arduino digital 9 pin
- Flame sensor GND pin connected to Arduino GND pin
- Flame sensor VCC pin connected to Arduino 3.3V pin
- Flame sensor A0 pin connected to Arduino analog A0 pin

You can see the wiring diagram here:



For example, you can see my wiring implementation on the Arduino Leonardo here:



To access the buzzer device and generate sound, we can use the `Tone` object. You can learn more about it from https://www.arduino.cc/en/Reference/Tone.

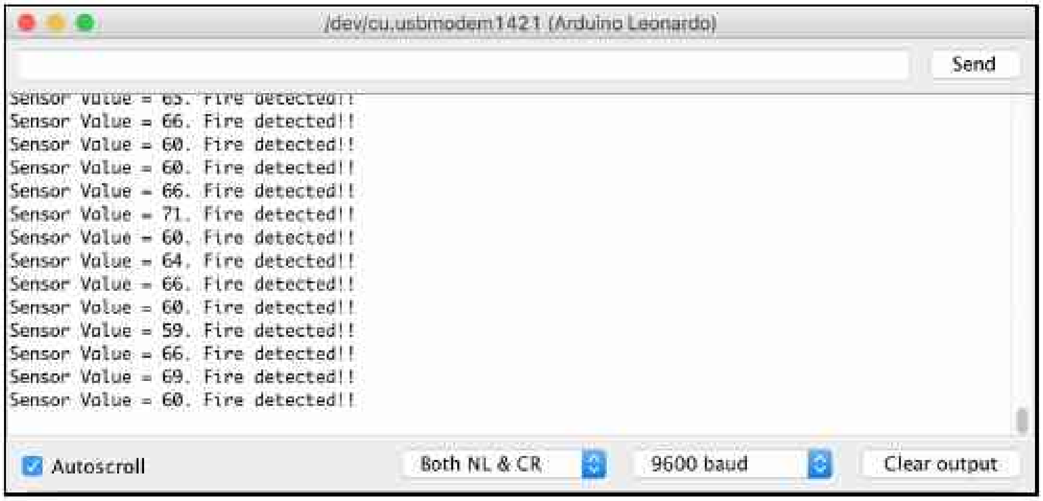
The algorithm implementation is easy. Firstly, we open the Arduino software and write the following sketch program:

```
int flameSensor = A0;
int buzzer = 9;
int val = 0;

void setup() {
  pinMode(buzzer,OUTPUT);
  Serial.begin(9600);
}

void loop() {
  val = analogRead(flameSensor);
  if(val > 50) {
    Serial.print("Sensor Value = ");
    Serial.print(val);
    Serial.println(". Fire detected!!");

    tone(buzzer,1000);
  }
  else
    noTone(buzzer);

  delay(500);
}
```

Save this sketch as `ArduinoFireDetection`.

Now you build and upload your sketch program to your Arduino board. After it's uploaded, you can open the Serial Monitor tool from Arduino to see the program output.

Try to move your sensor next to a flame so you can see the sensor value on the Serial Monitor tool. You can see a sample program output here:



```
Sensor Value = 65. Fire detected!!
Sensor Value = 66. Fire detected!!
Sensor Value = 60. Fire detected!!
Sensor Value = 60. Fire detected!!
Sensor Value = 66. Fire detected!!
Sensor Value = 71. Fire detected!!
Sensor Value = 60. Fire detected!!
Sensor Value = 64. Fire detected!!
Sensor Value = 66. Fire detected!!
Sensor Value = 60. Fire detected!!
Sensor Value = 59. Fire detected!!
Sensor Value = 66. Fire detected!!
Sensor Value = 69. Fire detected!!
Sensor Value = 60. Fire detected!!
```

How it works

This program initializes all required pins in the `setup()` function.

```
int flameSensor = A0;
int buzzer = 9;
int val = 0;

void setup() {
  pinMode(buzzer,OUTPUT);
  Serial.begin(9600);
}
```

In `loop()` function, we read the flame sensor value by calling the `analogRead()` function. In this case, we set the threshold value to `50`. If the reading is greater than `50`, we turn on our buzzer device by calling the `tone()` function:

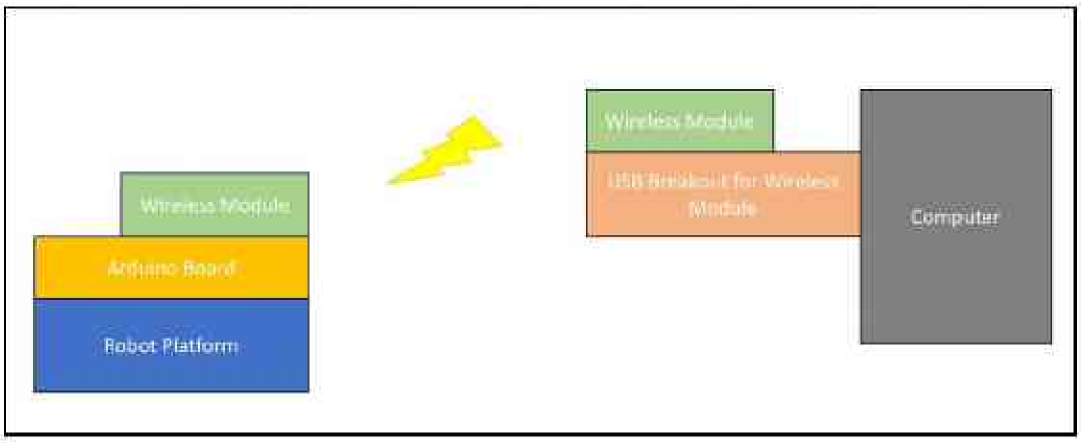
```
void loop() {
  val = analogRead(flameSensor);
  if(val > 50) {
    Serial.print("Sensor Value = ");
    Serial.print(val);
    Serial.println(". Fire detected!!");

    tone(buzzer,1000);
  }
  else
    noTone(buzzer);

  delay(500);
}
```

You can change the threshold value based on the results of your experiment.

Basic remote robot navigation

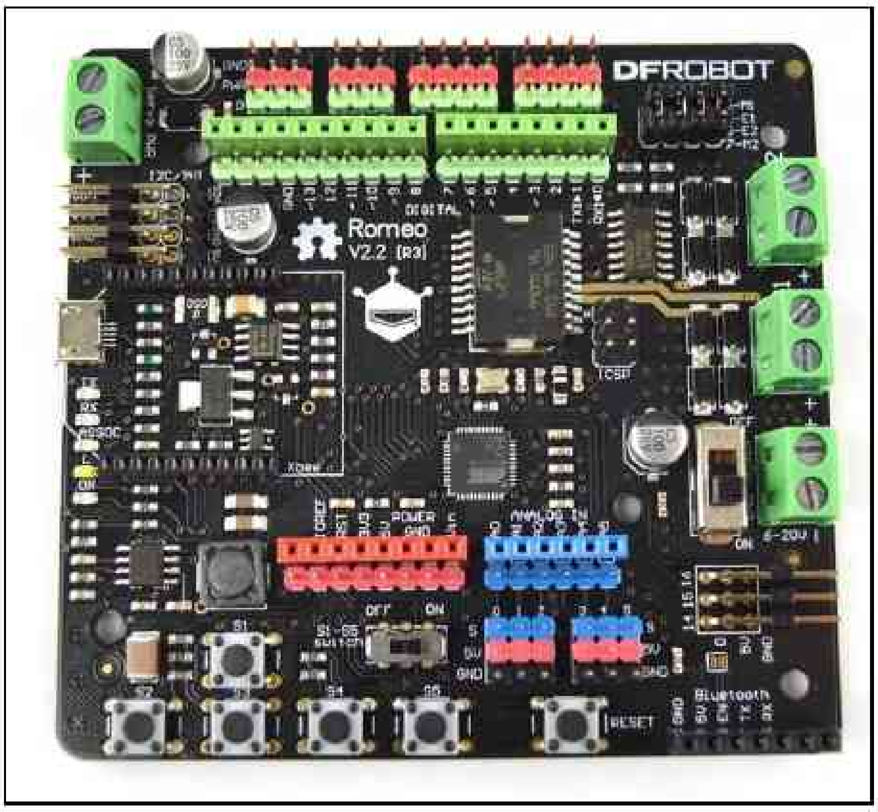
A robot can be controlled remotely through radio communication. We can use an RF module, Wi-Fi, or Bluetooth to communicate between robot and another system.

In this section, we'll try to control a robot from a computer using Bluetooth. In general, a communication model between a robot and a computer can be described as follows:



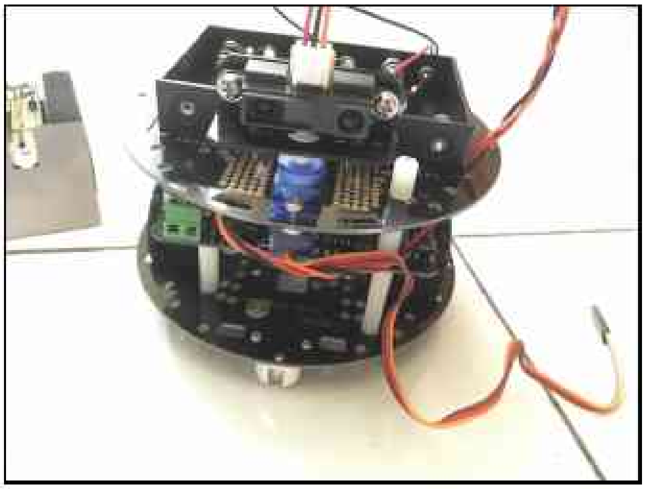
There are some radio and wireless modules that you can use to integrate with your robot to communicate with a computer.

For testing, I use the MiniQ Discover Arduino for robot platform from DFRobot: https://www.dfrobot.com/product-1144.html. Since this robot platform uses the Romeo board (Arduino compatible) as the core robot board, it supports an RF/wireless module-based XBee shield. You can check it out at https://www.dfrobot.com/product-844.html. This is what it looks like:



I use Bluno Bee (https://www.dfrobot.com/product-1073.html) from DFRobot for the RF module on the Romeo board. This module has a serial BLE network stack. You can attach it to the board and listen for incoming messages from serial pins. Bluno Bee has the same pin model as the XBee module. You can see it here:

Put the Bluno Bee module into the MiniQ Discover Arduino robot on the Romeo board. You also need another Bluno Bee that is attached to your computer through a micro-USB cable. You can see my implementation of the robot kit here:



Now we'll continue to develop a program to enable our robot to be controlled from a computer. Make sure you've turned off the robot. Using a micro-USB cable attached to the Romeo board, you can write a sketch program.

Open the Arduino software and write the following complete program:

```
//Standard PWM DC control
int E1 = 5;     //M1 Speed Control
int E2 = 6;     //M2 Speed Control
int M1 = 4;    //M1 Direction Control
int M2 = 7;    //M1 Direction Control

void stop(void)                    //Stop
{
  Serial.println("stop");
  digitalWrite(E1,LOW);
  digitalWrite(E2,LOW);
}
void advance(char a,char b)          //Move forward
{
  Serial.println("advance");
  analogWrite (E1,a);      //PWM Speed Control
  digitalWrite(M1,HIGH);
  analogWrite (E2,b);
  digitalWrite(M2,HIGH);
}
void back_off (char a,char b)          //Move backward
{
  Serial.println("back_off");
  analogWrite (E1,a);
  digitalWrite(M1,LOW);
  analogWrite (E2,b);
  digitalWrite(M2,LOW);
}
void turn_L (char a,char b)             //Turn Left
{
  Serial.println("turn_L");
  analogWrite (E1,a);
  digitalWrite(M1,LOW);
  analogWrite (E2,b);
  digitalWrite(M2,HIGH);
}
void turn_R (char a,char b)             //Turn Right
{
  Serial.println("turn_R");
  analogWrite (E1,a);
  digitalWrite(M1,HIGH);
  analogWrite (E2,b);
  digitalWrite(M2,LOW);
}
void setup(void)
{
  int i;
  for(i=4;i<=7;i++)
    pinMode(i, OUTPUT);
  Serial.begin(115200);      //Set Baud Rate
  Serial.println("Run keyboard control");

  Serial1.begin(115200);      //Set Baud Rate
}
void loop(void)
{

  if(Serial1.available()){
    char val = Serial1.read();
    if(val != -1)
    {
      switch(val)
      {
      case 'w'://Move Forward
        Serial.println("Move Forward");
        advance (100,100);   //move forward
        break;
      case 's'://Move Backward
        Serial.println("Move Backward");
        back_off (100,100);   //move back
        break;
      case 'a'://Turn Left
        Serial.println("Turn Left");
        turn_L (100,100);
        break;
      case 'd'://Turn Right
        Serial.println("Turn Right");
        turn_R (100,100);
        break;
      case 'z':
        Serial.println("Hello");
        break;
      case 'x':
        stop();
        break;
      }
    }
    else stop();
  }
}
```

Save this sketch as `ArduinoRobotDemo`. Now you can build and upload it to the Romeo board.

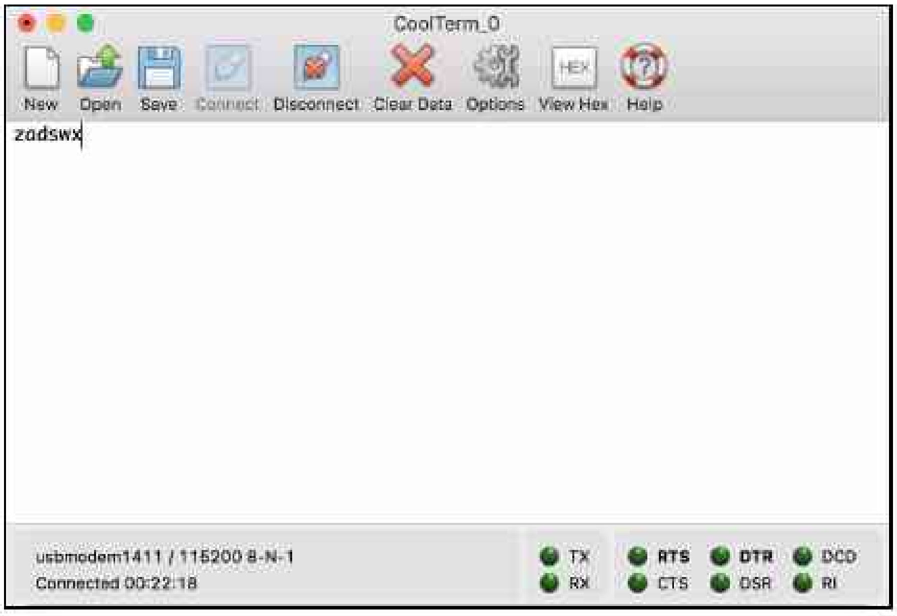
Now your robot should switch to battery power. Your robot should be ready to receive a command from the computer.

Since one Bluno Bee module is connected to a computer, you can use any serial tool to send messages to Bluno Bee. I use CoolTerm (http://freeware.the-meiers.org) for testing.

Connect the CoolTerm application to the serial port of Bluno Bee. By default, it uses a baud rate of 115200. Based on our sketch program, you can use these keys to move the robot:

- *W* to move forward
- *S* to move backward
- *A* to move left
- *D* to move right
- *Z* to print hello to the serial port
- *X* to stop

You can see my CoolTerm application output here:



```
zadswx
```

usbmodem1411 / 115200 8-N-1
Connected 00:22:18

Detecting obstacles

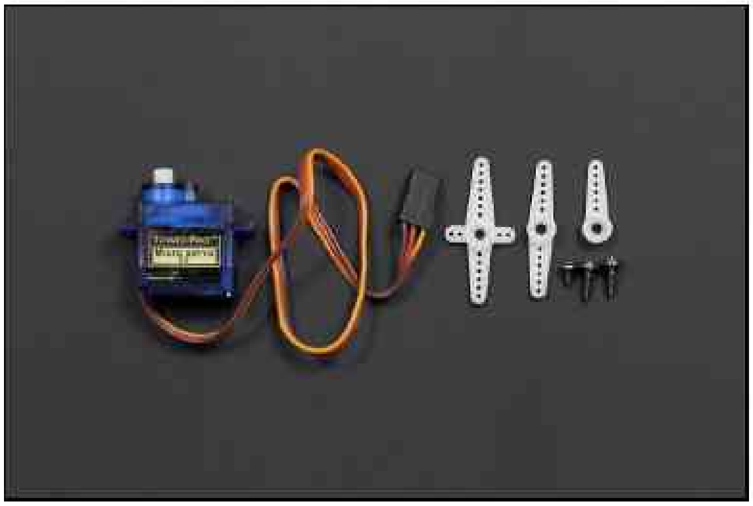
To move properly, our robot shouldn't be hindered by obstacles. A common solution to detect obstacles is to use an ultrasonic sensor. A sample sensor implementation is HC-SR04. You can get it from SparkFun: https://www.sparkfun.com/products/13959. You can see it here:

The HC-SR04 sensor has four pins: VCC (Power), Trig (Trigger), Echo (Receive), and GND (Ground), which are easy to connect to Arduino and Raspberry Pi.

Another sensor is the IR distance sensor. A sample sensor implementation is the Sharp GP2Y0A21 IR Distance Sensor, https://www.dfrobot.com/product-328.html, from DFRobot. Since I'm using the MiniQ Discovery Arduino robot, this sensor can be attached directly to the robot shield. You can see this here:

The GP2Y0A21 sensor has three pins: VCC, GND, and Signal. The Signal pin of GP2Y0A21 can be connected to analog pins from the Arduino board.

We can also scan obstacles around the robot. In this case, we need a servo motor and attach our sensor to the servo motor. You can get it from DFRobot on this site: https://www.dfrobot.com/product-255.html. You can also can see this kit here:

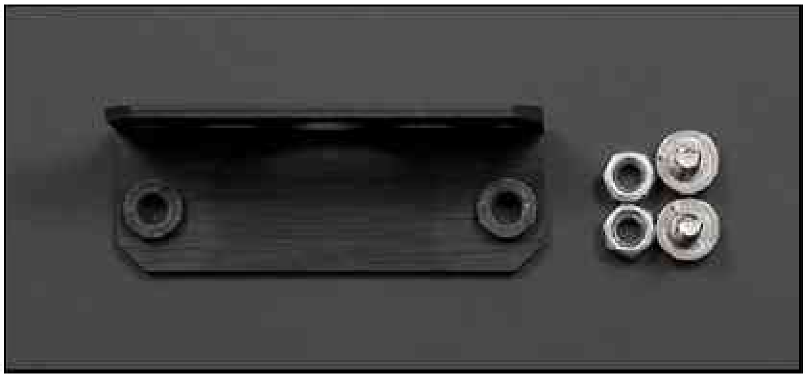


In general, a servo has three wires with the following pinout information:

- Brown wire is GND
- Red wire is VCC
- Orange wire is the signal line

You can connect a servo to Arduino through PWM pins.

To attach the Sharp GP2Y0A21 IR distance sensor to a servo motor, you need a mounting bracket like this product (https://www.dfrobot.com/product-127.html):

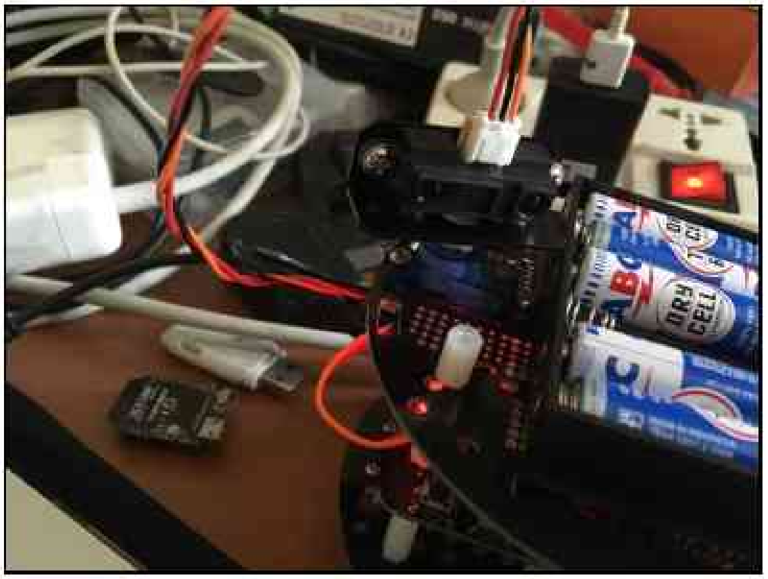


DFRobot also provides a complete kit, which consists of a Sharp GP2Y0A21 sensor, servo motor, and mounting bracket. You can check it out on https://www.dfrobot.com/product-715.html.

For this demo, we will wire it as follows:

- GP2Y0A21 VCC to Arduino 3.3V
- GP2Y0A21 GND to Arduino GND
- GP2Y0A21 Signal to Arduino analog pin A0
- Servo VCC to Arduino 5V
- Servo GND to Arduino GND
- Servo Signal to Arduino digital pin 9

After you've attached the sensor, mounting, and servo motor to the robot, you can start developing your sketch program. You can see my implementation here:



In the sketch program, we can access the servo motor using the Servo library. For further information on the Servo library, I recommend you read https://www.arduino.cc/en/Reference/Servo.

We will develop a sketch program to scan the obstacle distance around a robot. We'll move our sensor from 0 to 180 degrees through the servo motor.

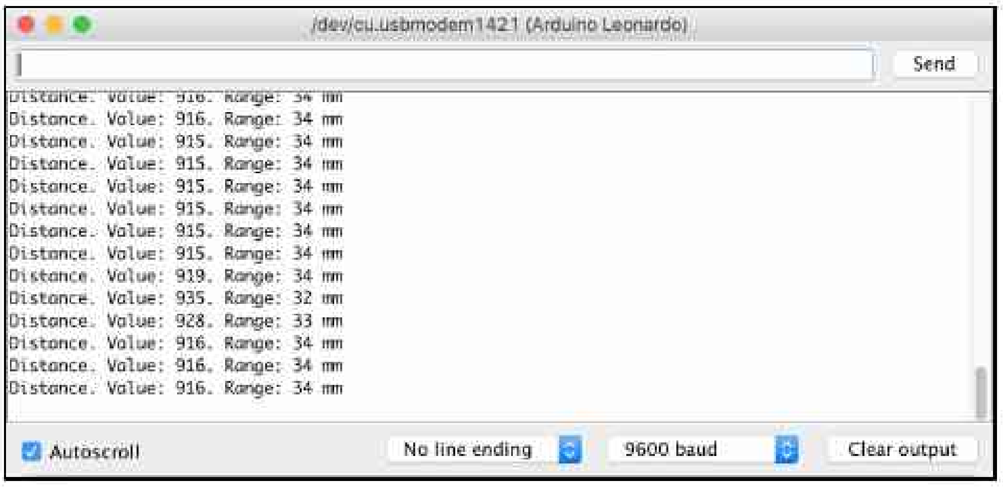
Now you can open the Arduino software and write this sketch program:

```
#include <Servo.h>

Servo servo;
#define Svo_Pin   9
#define GP2Y0A21  A0

void setup() {
  servo.attach(Svo_Pin);
  Serial.begin(9600);
}

void loop() {
  for (int i=0;i<180;i++)
  {
    servo.write(i);
    uint16_t value = analogRead(GP2Y0A21);
    uint16_t range = get_gp2d12(value);
    Serial.print("Distance. Value: ");
    Serial.print(value);
    Serial.print(". Range: ");
    Serial.print(range);
    Serial.println(" mm");

    delay(200);
  }
  delay(2000);
}
```

```
uint16_t get_gp2d12 (uint16_t value) {
  if (value < 10) value = 10;
  return ((67870.0 / (value - 3.0)) - 40.0);
}
```

Save this sketch program as `ArduinoDisctanceScanner`.

Build and upload the program to your Arduino board. You can open the Serial Monitor tool from Arduino to see the following program output:



```
Distance. Value: 916. Range: 34 mm
Distance. Value: 916. Range: 34 mm
Distance. Value: 915. Range: 34 mm
Distance. Value: 915. Range: 34 mm
Distance. Value: 915. Range: 34 mm
Distance. Value: 915. Range: 34 mm
Distance. Value: 915. Range: 34 mm
Distance. Value: 915. Range: 34 mm
Distance. Value: 919. Range: 34 mm
Distance. Value: 935. Range: 32 mm
Distance. Value: 928. Range: 33 mm
Distance. Value: 916. Range: 34 mm
Distance. Value: 916. Range: 34 mm
Distance. Value: 916. Range: 34 mm
```

```
#include <Servo.h>

Servo servo;

#define Svo_Pin 9

#define GP2Y0A21 A0



void setup() {

   servo.attach(Svo_Pin); Serial.begin(9600); }

void loop() {

   for (int i=0;i<180;i++) {

   servo.write(i); uint16_t value = analogRead(GP2Y0A21); uint16_t
range = get_gp2d12(value); Serial.print("Distance. Value: ");
Serial.print(value); Serial.print(". Range: "); Serial.print(range);
Serial.println(" mm");

   delay(200); }

   delay(2000); }

uint16_t get_gp2d12 (uint16_t value) {

   if (value < 10) value = 10; return ((67870.0 / (value - 3.0)) - 40.0); }
```

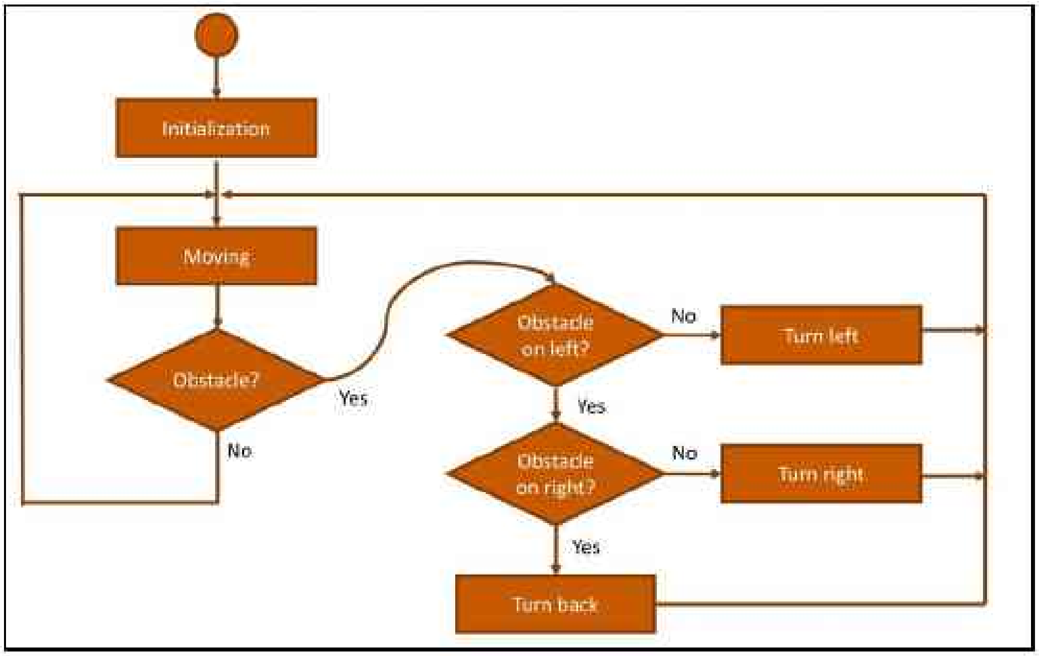
Designing an autonomous robot

Designing and implementing an autonomous robot needs a lot of work. The biggest challenge of an autonomous robot is being movement and navigation and how to achieve objectives.

An autonomous robot runs on its own accord. This is challenging because we need to make our robot decide how it moves. From an artificial intelligence perspective, we can classify two models: supervised and unsupervised.

A supervised autonomous robot requires some objectives or guidelines to achieve its goals. On the other hand, an unsupervised autonomous robot can learn from its experiences and then make own decision based on certain conditions.

To build an autonomous robot for movement and navigation, we can design our algorithm as follows:



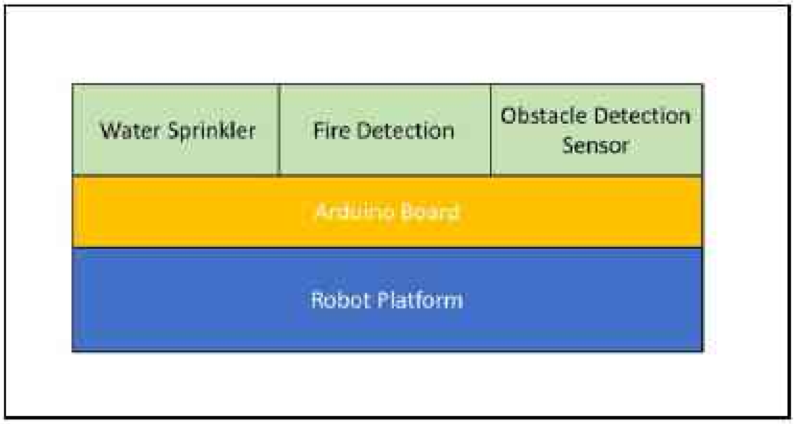
Before starting, a robot will initialize all parameters included for its sensors and motors. Then, we'll start by moving it in a random direction. While moving, the robot may detect an obstacle. If one is found, the robot will change direction.

In this scenario, we aren't defining the the goal of the autonomous robot. An example goal could be if it finds a star sign, the robot should stop and power off. You can define your own decisions and goals.

Building an autonomous firefighter robot

An autonomous firefighter robot is basically an autonomous robot with an objective of sprinkling water on a fire. To achieve this goal, the robot will move around to find a fire source. While moving, a robot should handle obstacle problems.

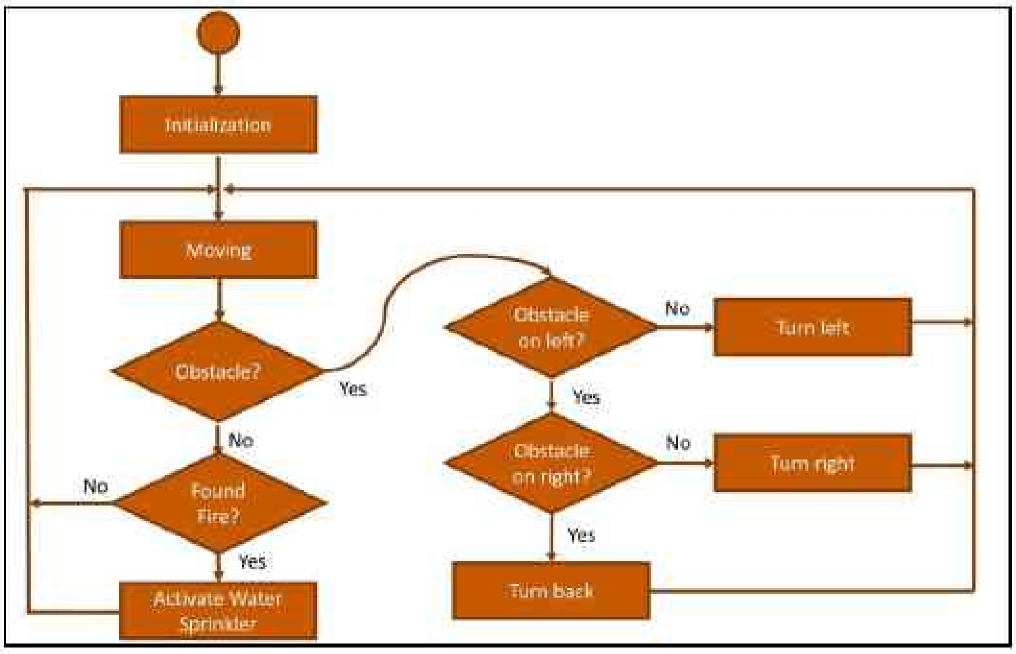
A sample design of an autonomous firefighter robot can be seen here:



The robot should have a water sprinkler module and fire and obstacle detection sensors. The robot platform can be based in Arduino, Raspberry Pi, or other MCU boards.

The idea of an autonomous robot for firefighting is similar in behavior to a common autonomous robot, but with additional features to detect fires and sprinkle water on them.

We can modify our autonomous robot design from the previous section for a firefighter robot scenario. You can see the modified design here:



We've added fire detection and water sprinkling to our algorithm. When the robot is moving, it should detect obstacles and fire. If it finds a fire, it should sprinkle water on it to put it out.

Summary

We learned about robot platforms in this chapter. Then, we learned about firefighter robots that move and detect a fire source and obstacles. Lastly, we design an autonomous robot for firefighting.

In the next chapter, we will explore multi-robot cooperation using swarm intelligence.

Multi-Robot Cooperation Using Swarm Intelligence

Coordinating autonomous robots is challenging: each robot has its own intelligence system. These robots try to perform self-managed activities to achieve a particular goal. In this chapter, we'll learn how to work and deal with multiple robots in order to execute their goals.

We will learn about the following topics:

- Introducing multi-robot cooperation
- Learning about swarm intelligence
- Implementing a mesh network for a multi-robot scenario
- XBee development for Arduino
- Designing a multi-robot cooperation model using swarm intelligence

Let's start!

Introducing multi-robot cooperation

Communicating and negotiating among robots is challenging. We should ensure our robots address collision while they are moving. Meanwhile, these robots should achieve their goals collectively.

For example, Keisuke Uto has created a multi-robot implementation to create a specific formation. They take input from their cameras. Then, these robots arrange themselves to create a formation. To get the correct robot formation, this system uses a camera to detect the current robot formation. Each robot has been labelled so it makes the system able to identify the robot formation.

By implementing image processing, Keisuke shows how multiple robots create a formation using multi-robot cooperation. If you are interested, you can read about the project at https://www.digi.com/blog/xbee/multi-robot-formation-control-by-self-made-robots/. Here they are in action:

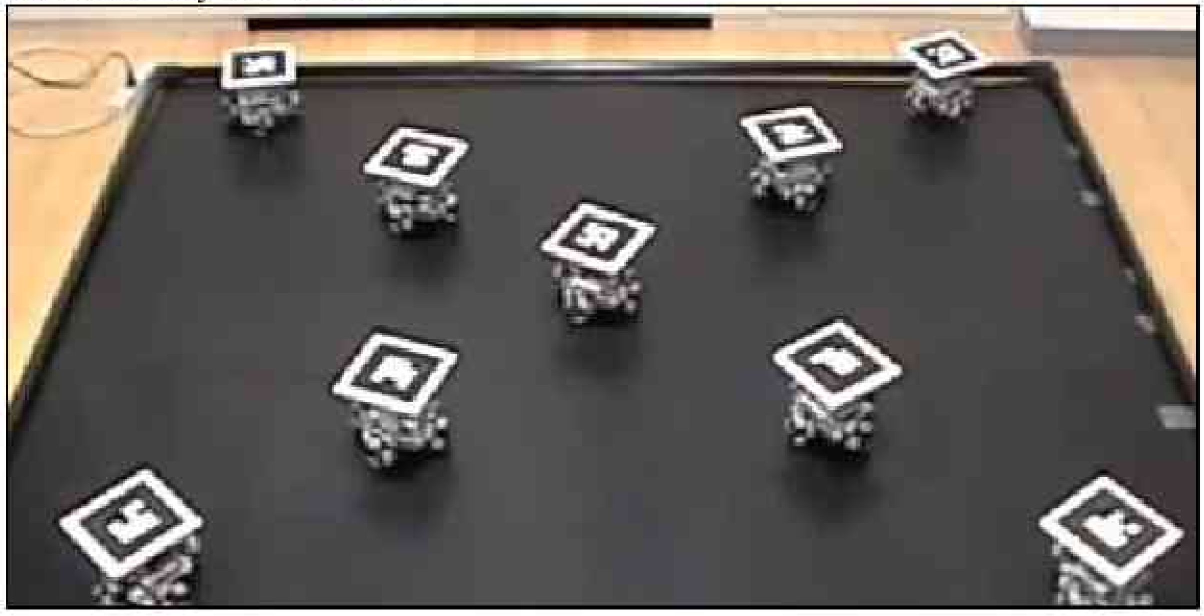


Image source: https://www.digi.com/blog/xbee/multi-robot-formation-control-by-self-made-robots/

For another example, can have soccer matches in which the players are not human; they are all robots. Such a competition can use custom robots or existing commercial robots such as the Nao robot, https://www.ald.softbankrobotics.com/en/robots/nao. You can see Nao robots in a soccer match here:

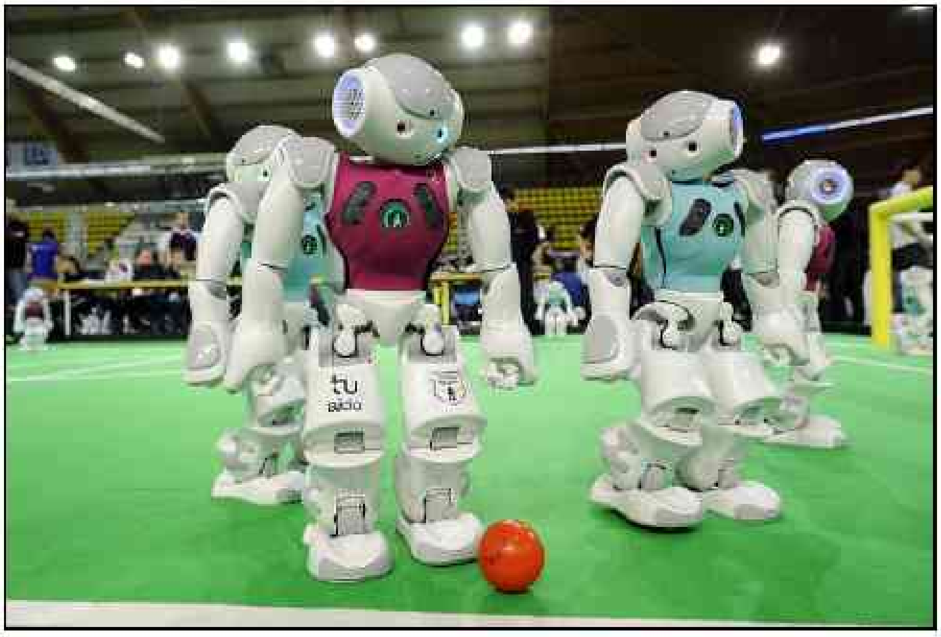


Image source: http://www.robocup2014.org/?p=893

As we know, a robot soccer game involves robots, and they need skill and knowledge in order to win. The robot should detect the ball and score goals. Meanwhile, each robot should perform group cooperation. Addressing collision among robots and passing the ball to its group are challenging in this area.

Learning about swarm intelligence

Swarm intelligence is inspired by the collective behavior of social animal colonies such as ants, birds, wasps, and honey bees. These animals work together to achieve a common goal.

Swarm intelligence phenomena can be found in our environment. You can see swarm intelligence in deep-sea animals, shown in the following image of a school of fish in formation that was captured by a photographer in Cabo Pulmo:

Image source: http://octavioaburto.com/cabo-pulmo

Using information from swarm intelligence studies, swarm intelligence is applied to coordinate among autonomous robots. Each robot can be described as a self-organization system. Each one negotiates with the others on how to achieve the goal.

There are various algorithms to implement swarm intelligence. I recommend you read textbooks about it. Some math and statistics are applied for implementing swarm intelligence. The following is a list of swarm intelligence types that researchers and developers apply to their problems:

- Particle swarm optimization

- Ant system
- Ant colony system
- Bees algorithm
- Bacterial foraging optimization algorithm

The **Particle Swarm Optimization** (**PSO**) algorithm is inspired by the social foraging behavior of some animals such as the flocking behavior of birds and the schooling behavior of fish. A sample of PSO algorithm in Python can be found at https://gist.github.com/btbytes/79877. This program needs the `numpy` library. `numpy` (Numerical Python) is a package for scientific computing with Python. You computer should has installed Python. If not, you can download and install on this site, https://www.python.org. If your computer does not have `numpy` , you can install it by typing this command in the terminal (Linux and Mac platforms): **$ pip install numpy**

For Windows platform, please install `numpy` refer to this https://www.scipy.org/install.html.

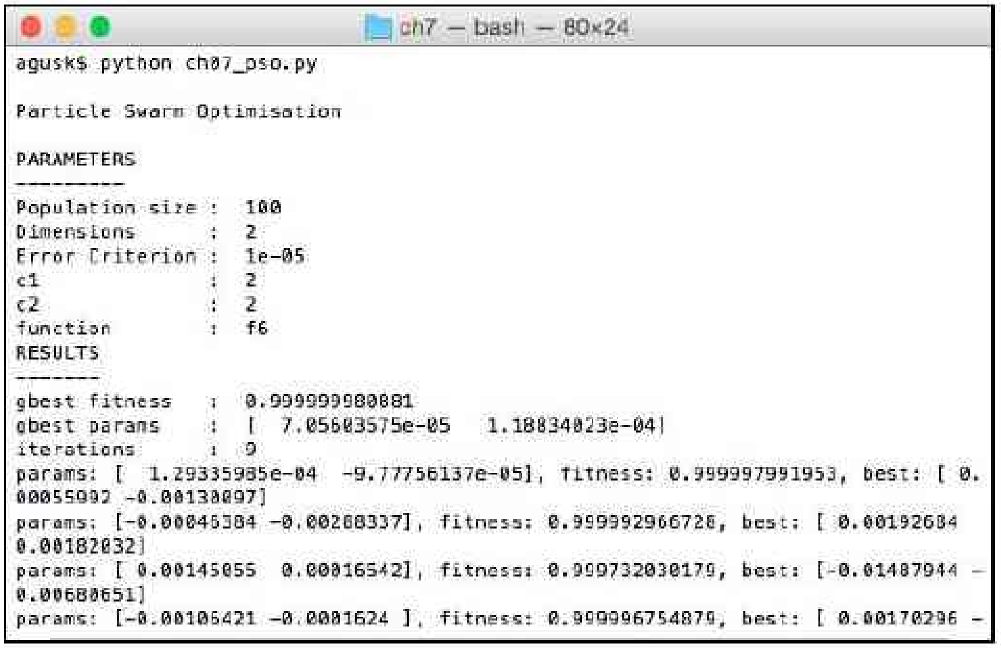
You can copy the following code into your editor. Save it as `ch07_pso.py` and then run it on your computer using terminal:

```
from numpy import array
from random import random
from math import sin, sqrt

iter_max = 10000
pop_size = 100
dimensions = 2
c1 = 2
c2 = 2
err_crit = 0.00001

class Particle:
    pass


def f6(param):
    '''Schaffer's F6 function'''
    para = param*10
    para = param[0:2]
    num = (sin(sqrt((para[0] * para[0]) + (para[1] * para[1])))) * \
        (sin(sqrt((para[0] * para[0]) + (para[1] * para[1])))) - 0.5
    denom = (1.0 + 0.001 * ((para[0] * para[0]) + (para[1] * para[1]))) * \
            (1.0 + 0.001 * ((para[0] * para[0]) + (para[1] * para[1])))
    f6 =  0.5 - (num/denom)
    errorf6 = 1 - f6
    return f6, errorf6;


#initialize the particles
particles = []
for i in range(pop_size):
    p = Particle()
    p.params = array([random() for i in range(dimensions)])
    p.fitness = 0.0
    p.v = 0.0
    particles.append(p)

# let the first particle be the global best
gbest = particles[0]
err = 999999999
while i < iter_max :
    for p in particles:
        fitness,err = f6(p.params)
        if fitness > p.fitness:
            p.fitness = fitness
            p.best = p.params

        if fitness > gbest.fitness:
            gbest = p
        v = p.v + c1 * random() * (p.best - p.params) \
                + c2 * random() * (gbest.params - p.params)
        p.params = p.params + v

    i  += 1
    if err < err_crit:
        break
    #progress bar. '.' = 10%
    if i % (iter_max/10) == 0:
        print '.'

print '\nParticle Swarm Optimisation\n'
print 'PARAMETERS\n','-'*9
print 'Population size : ', pop_size
print 'Dimensions : ', dimensions
print 'Error Criterion : ', err_crit
print 'c1 : ', c1 print 'c2 : ', c2
print 'function : f6'

print 'RESULTS\n', '-'*7
print 'gbest fitness   : ', gbest.fitness
print 'gbest params    : ', gbest.params
print 'iterations      : ', i+1

## Uncomment to print particles
for p in particles:
   print 'params: %s, fitness: %s, best: %s' % (p.params, p.fitness, p.best)
```

You can run this program by typing this command:

```
$ python ch07_pso.py
```

You can see a sample of my program output here:



```
agusk$ python ch07_pso.py

Particle Swarm Optimisation

PARAMETERS
----------
Population size :  100
Dimensions      :  2
Error Criterion :  1e-05
c1              :  2
c2              :  2
function        :  f6
RESULTS
--------
gbest fitness   :  0.999999980881
gbest params    :  [  7.05603575e-05   1.18834023e-04]
iterations      :  9
params: [  1.29335985e-04  -9.77756137e-05], fitness: 0.999997991953, best: [ 0.
00055992 -0.00138097]
params: [-0.00046384 -0.00288337], fitness: 0.999992966728, best: [ 0.00192684
0.00182032]
params: [ 0.00145055  0.00016542], fitness: 0.999732030179, best: [-0.01487944 -
0.00680651]
params: [-0.00106421 -0.0001624 ], fitness: 0.999996754879, best: [ 0.00170296 -
```

This program will generate PSO output parameters based on input. You can see PARAMETERS value on program output. At the end of codes, we can print all PSO particle parameter while iteration process.

Implementing mesh network for multi-robot cooperation

To implement multi-robot cooperation, we need a communication medium. In general, RF communication is chosen by researchers and makers to exchange data among robots. In this book, we will use XBee modules from Digi International to establish RF communication.

XBee modules

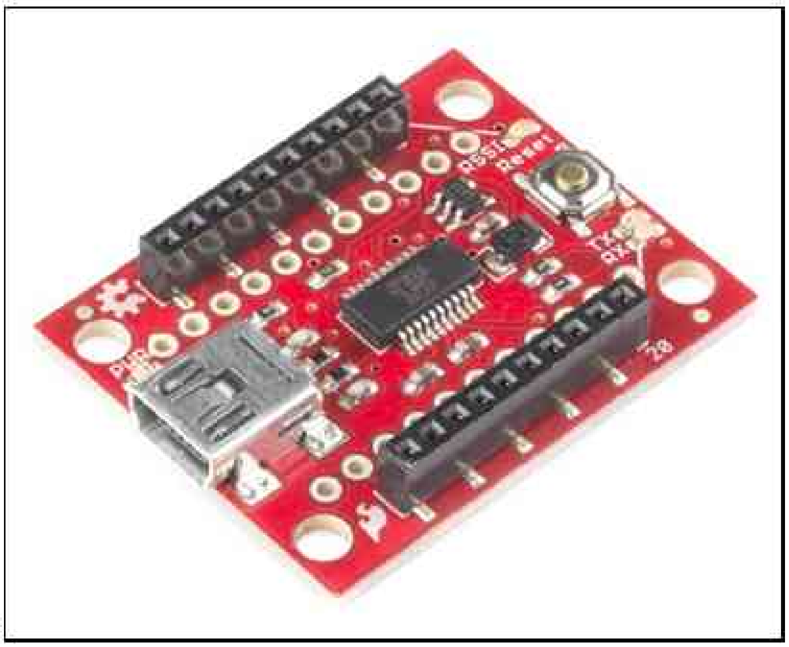
XBee modules are RF modules that are manufactured by Digi International. There are various XBee modules that we can use in our robots. One of the XBee modules is the XBee Series 1 that supports the IEEE 802.15.4 protocol. This module has features to address multi-point packets. For further information about the XBee Series 1, you can visit this site: https://www.digi.com/products/xbee-rf-solutions/2-4-ghz-modules/xbee-802-15-4.

XBee modules can be found in several online stores and even in your local stores. The following image is of the XBee Pro 60mW PCB Antenna - Series 1 that is available on SparkFun: https://www.sparkfun.com/products/11216. Another product model that can also be used is the XBee 1mW Trace Antenna - Series 1 (https://www.sparkfun.com/products/11215).

Basically, the XBee Series 1 supports point-to-point networks but we can configure it using DigiMesh firmware in order to work with a mesh network. I will explain how to configure the XBee Series 1 to use DigiMesh firmware in the next section.

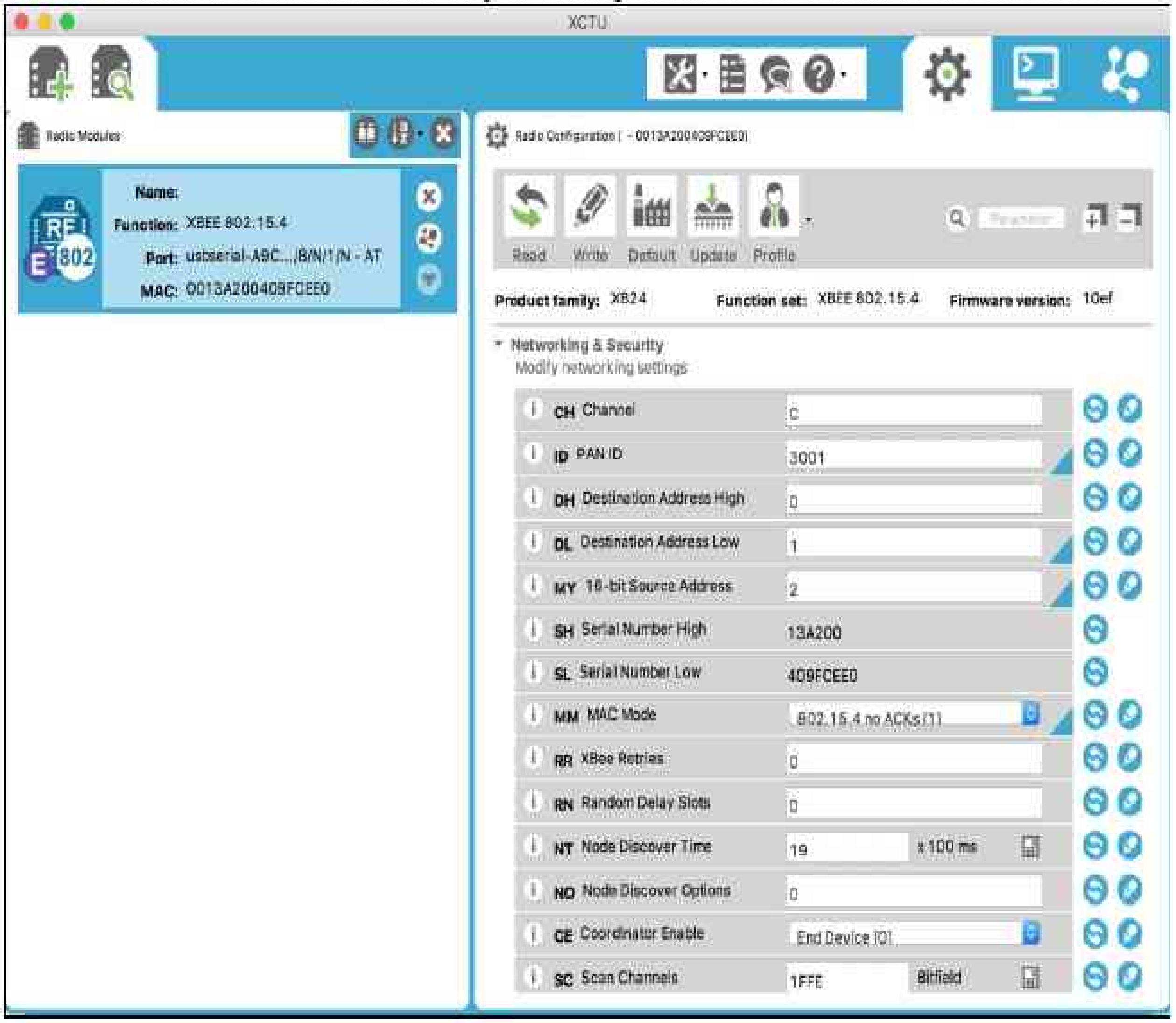
Another option is you can buy the XBee DigiMesh and XBee S2C or the latest of XBee Series 2 module to work with mesh networks. For further information about the XBee DigiMesh, you can read https://www.digi.com/products/xbee-rf-solutions/2-4-

ghz-modules/xbee-digimesh-2-4. Here is what the XBee S2C looks like:

These XBee modules have their own header pins so if we want to access these pins, we'll need an XBee shield or breakout. There are many XBee shields that fit your board. They are also available for use through PCs via XBee USB, for example, SparkFun XBee Explorer USB: https://www.sparkfun.com/products/11812. You can put an XBee module into this breakout. Then, you can connect it to your computer through a USB cable. You can see the SparkFun XBee Explorer USB here: 

Configuring XBee modules

Digi has provided a software tool to configure all XBee modules. It's called XCTU. It can be downloaded from https://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu. Download the version for your OS platform. Here's what it looks like:



Radio Configuration [- 0013A200409FCEE0]

Name:
Function: XBEE 802.15.4
Port: usbserial-A9C.../8/N/1/N - AT
MAC: 0013A200409FCEE0

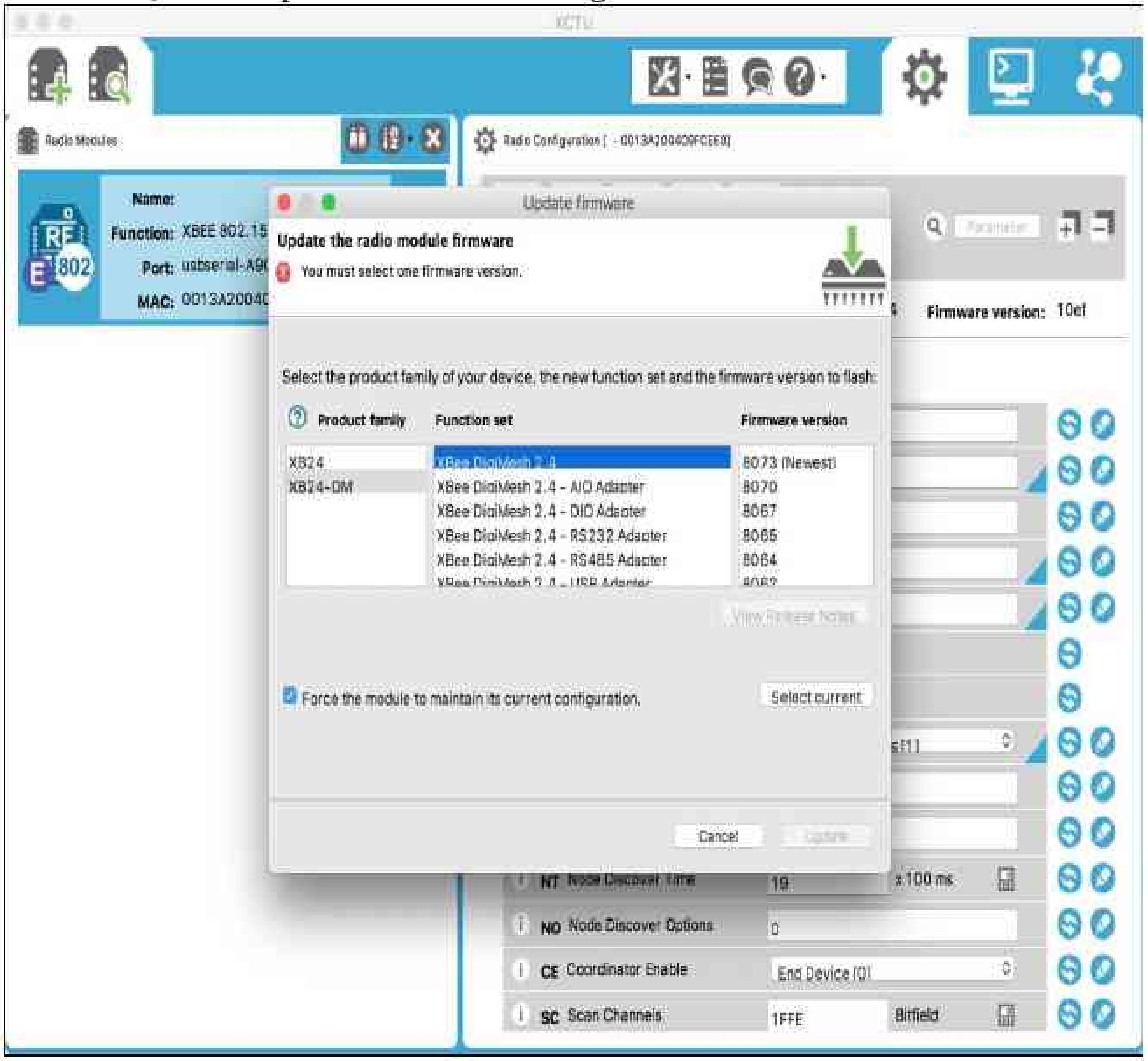
Product family: XB24 Function set: XBEE 802.15.4 Firmware version: 10ef

Networking & Security
Modify networking settings

CH	Channel	C
ID	PAN ID	3001
DH	Destination Address High	0
DL	Destination Address Low	1
MY	16-bit Source Address	2
SH	Serial Number High	13A200
SL	Serial Number Low	409FCEE0
MM	MAC Mode	802.15.4 no ACKs [1]
RR	XBee Retries	0
RN	Random Delay Slots	0
NT	Node Discover Time	19 x 100 ms
NO	Node Discover Options	0
CE	Coordinator Enable	End Device [0]
SC	Scan Channels	1FFE Bitfield

Make sure all XBee modules have installed the XBee Digimesh firmware in order to build a mesh network.

After the XBee module is attached to an XBee breakout USB and connected to a computer, you can add it to the XCTU application. If you have an XBee series 1, you should update your XBee modules with XBee Digimesh. Select the latest firmware for the XB24-DM firmware model, as shown in the following screenshot, on the updated firmware dialog:


Update the radio module firmware

You must select one firmware version.

Select the product family of your device, the new function set and the firmware version to flash:

Product family	Function set	Firmware version
XB24	XBee DigiMesh 2.4	8073 (Newest)
XB24-DM	XBee DigiMesh 2.4 - AIO Adapter	8070
	XBee DigiMesh 2.4 - DIO Adapter	8067
	XBee DigiMesh 2.4 - RS232 Adapter	8065
	XBee DigiMesh 2.4 - RS485 Adapter	8064

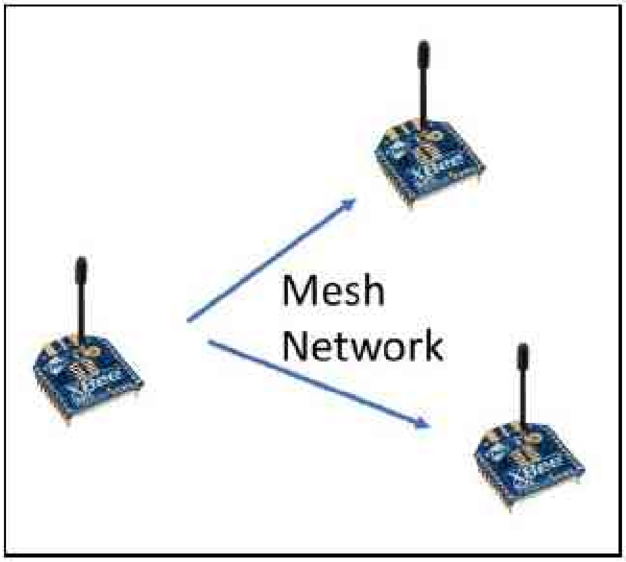
Force the module to maintain its current configuration.

XCTU
Radio Modules
Radio Configuration [- 0013A20040FCE60]
Name:
Function: XBEE 802.15
Port: usbserial-A9
MAC: 0013A20040
Firmware version: 10ef
View Release Notes
Select current
Cancel
Update
NT Node Discover Time 19 x 100 ms
NO Node Discover Options 0
CE Coordinator Enable End Device [0]
SC Scan Channels 1FFE Bitfield

Installing Digimesh firmware on XBee modules has its benefits. We don't need a coordinator in a network. We only need to define our network ID for network identity.

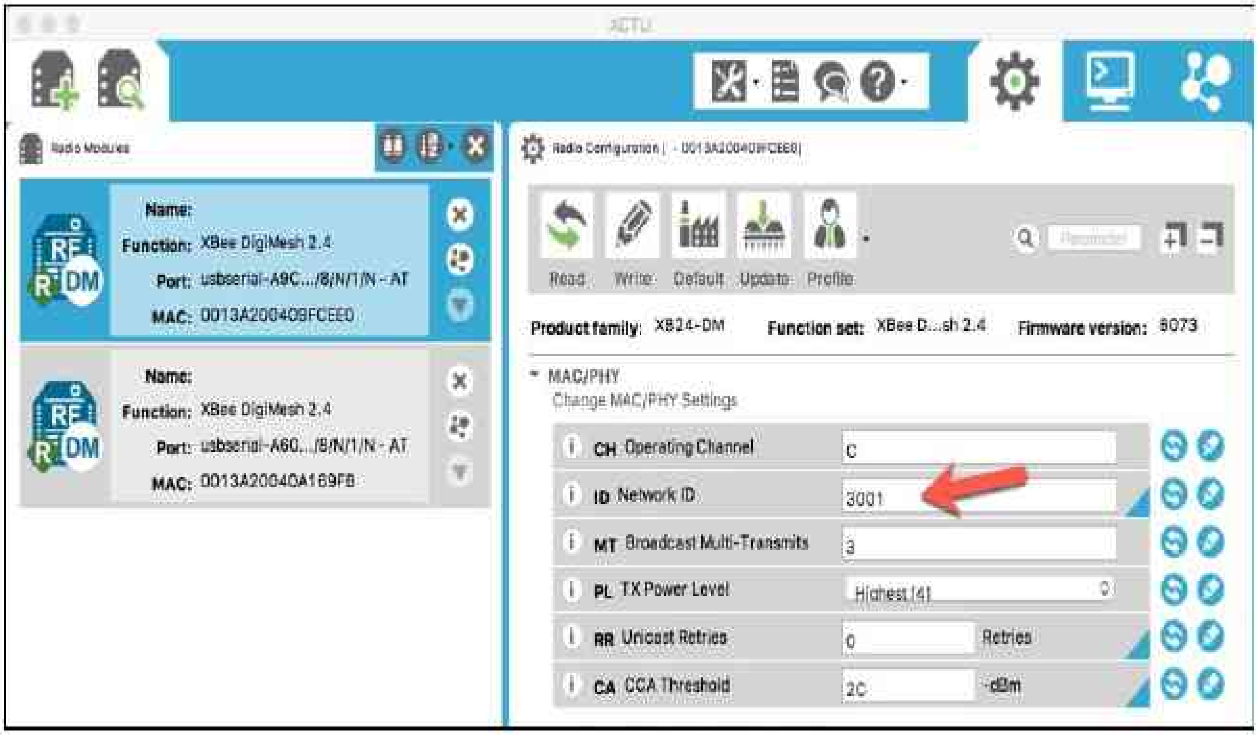
Demo - data communication using XBee

In this section, we'll develop a Python application to show how to communicate between XBee modules. We need two XBee modules or more to implement our demo. We will implement a mesh network like the following for our demo:



All XBee modules should already have XBee Digimesh firmware installed. Now we'll configure our mesh network.

First, we define a network ID for our mesh network. We can do this using XCTU. You can open each XBee module in the XCTU application. Then, we'll set a network ID (ID) in XCTU:



Name:
Function: XBee DigiMesh 2.4
Port: usbserial-A9C.../8/N/1/N - AT
MAC: 0013A200409FCEE0

Name:
Function: XBee DigiMesh 2.4
Port: usbserial-A60.../8/N/1/N - AT
MAC: 0013A20040A169FB

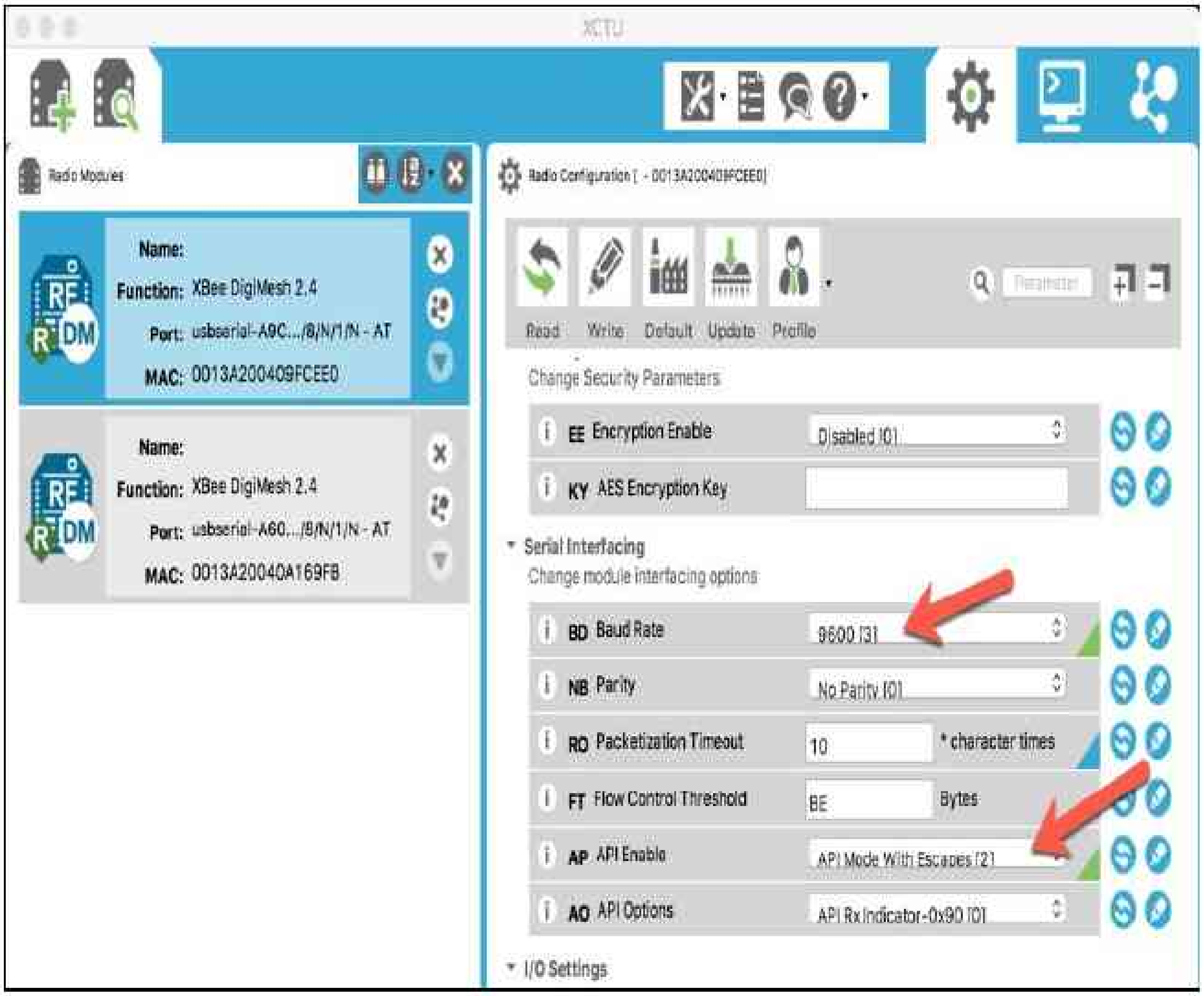
Radio Configuration [- 0013A200409FCEE0]

Product family: XB24-DM Function set: XBee D...sh 2.4 Firmware version: 8073

MAC/PHY
Change MAC/PHY Settings

CH	Operating Channel	C
ID	Network ID	3001
MT	Broadcast Multi-Transmits	3
PL	TX Power Level	Highest [4]
RR	Unicast Retries	0 Retries
CA	CCA Threshold	2C -dBm

Further, you also need to configure the **Baud Rate (BD)** as `9600 [3]` and set the API Enable value to API Mode with Escapes [2]. These settings can be seen here:



Name:
Function: XBee DigiMesh 2.4
Port: usbserial-A9C.../8/N/1/N - AT
MAC: 0013A200409FCEE0

Name:
Function: XBee DigiMesh 2.4
Port: usbserial-A60.../8/N/1/N - AT
MAC: 0013A20040A169FB

Radio Configuration [- 0013A200409FCEE0]

Change Security Parameters

EE	Encryption Enable	Disabled [0]
KY	AES Encryption Key	

Serial Interfacing
Change module interfacing options

BD	Baud Rate	9600 [3]
NB	Parity	No Parity [0]
RO	Packetization Timeout	10 * character times
FT	Flow Control Threshold	BE Bytes
AP	API Enable	API Mode With Escapes [2]
AO	API Options	API Rx Indicator-0x90 [0]

I/O Settings

After changing these XBee parameters in XCTU, you can write these changes to the XBee module by clicking on the Write icon. Close XCTU if you are done.

Now we can develop our Python application. We'll use the `xbee` library for Python. You can find it at https://pypi.python.org/pypi/XBee. To install this library on your computer, you can use pip. Type this command:

```
$ pip install xbee
```

We'll develop two programs, an XBee reader and an XBee sender.

The XBee reader application will listen for all incoming messages. Once the XBee module receives a message, the application will print the message to terminal. Create a Python file called `ch07_xbee_reader.py` and paste this program into it:

```
import time
from xbee import DigiMesh
import serial

#PORT = 'COM8'
PORT = '/dev/cu.usbserial-A601F21U'
BAUD_RATE = 9600


def ByteToHex(byteStr):
    return ''.join(["%02X" % ord(x) for x in byteStr]).strip()


def decodeReceivedFrame(data):
    source_addr = ByteToHex(data['source_addr'])
    rf_data = data['data']
    options = ByteToHex(data['options'])
    return [source_addr, rf_data, options]


# Open serial port
ser = serial.Serial(PORT, BAUD_RATE)

# Create API object
xbee = DigiMesh(ser, escaped=True)
import pprint
pprint.pprint(xbee.api_commands)

while True:
    try:
        data = xbee.wait_read_frame()
        decodedData = decodeReceivedFrame(data)
        print(decodedData)

    except KeyboardInterrupt:
        break

xbee.halt()
ser.close()
```

Change the PORT value based on your attached XBee serial port.

The next step is to develop the XBee sender application. For our demo, we'll send the messages `Hello XBee 2` to all XBee modules on the mesh network with a certain ID. You can write the following program:

```
import time
from xbee import DigiMesh
import serial

#PORT = 'COM7'
PORT = '/dev/cu.usbserial-A9CNVHXX'
BAUD_RATE = 9600

# Open serial port
ser = serial.Serial(PORT, BAUD_RATE)

# Create API object
xbee = DigiMesh(ser, escaped=True)
import pprint
```

```
pprint.pprint(xbee.api_commands)

# xbee with long address
XBEE2_ADDR_LONG = "\x00\x00\x00\x00\x00\x00\xFF\xFF"

while True:
    try:
        print "send data"
        xbee.tx(frame='0x1', dest_addr=XBEE2_ADDR_LONG, data='Hello XBee 2')
        time.sleep(1)
    except KeyboardInterrupt:
        break

xbee.halt()
ser.close()
```

Change the PORT value based on your XBee serial port. Save this program into a file called `ch07_xbee_sender.py`.

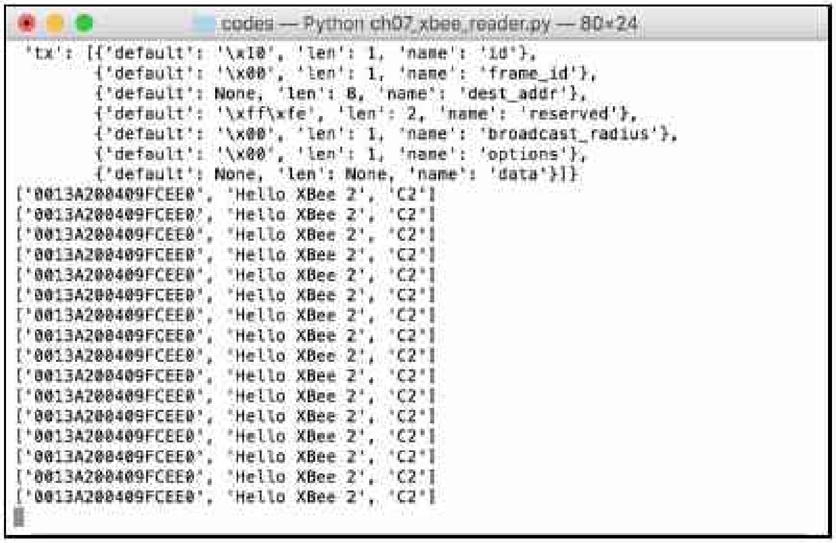
Now you can start running your XBee reader application. Type this on a computer to which the XBee module for the reader application is attached:

```
$ python ch07_xbee_reader.py
```

Then, run the XBee sender application by typing this command:

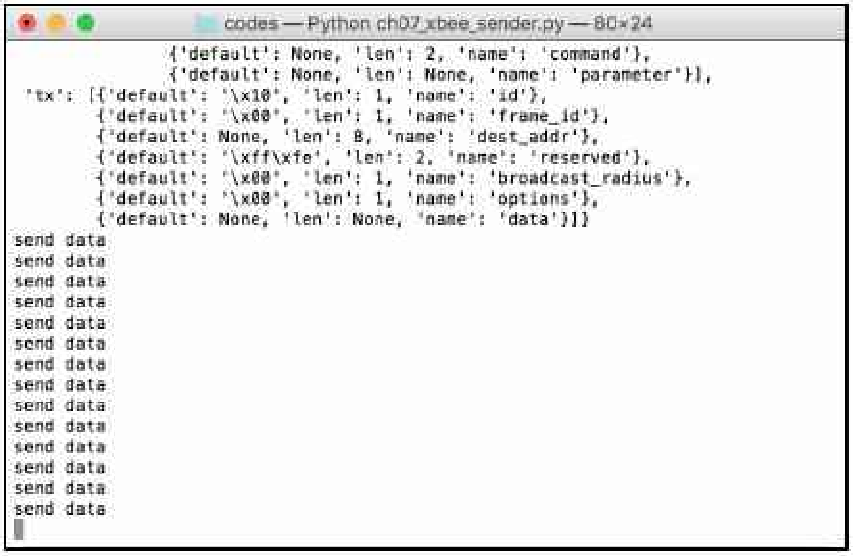
```
$ python ch07_xbee_sender.py
```

You should see incoming messages in the XBee reader application, shown here:



```
 'tx': [{'default': '\x10', 'len': 1, 'name': 'id'},
        {'default': '\x00', 'len': 1, 'name': 'frame_id'},
        {'default': None, 'len': 8, 'name': 'dest_addr'},
        {'default': '\xff\xfe', 'len': 2, 'name': 'reserved'},
        {'default': '\x00', 'len': 1, 'name': 'broadcast_radius'},
        {'default': '\x00', 'len': 1, 'name': 'options'},
        {'default': None, 'len': None, 'name': 'data'}]}
['0013A200409FCEE0', 'Hello XBee 2', 'C2']
['0013A200409FCEE0', 'Hello XBee 2', 'C2']
['0013A200409FCEE0', 'Hello XBee 2', 'C2']
['0013A200409FCEE0', 'Hello XBee 2', 'C2']
['0013A200409FCEE0', 'Hello XBee 2', 'C2']
['0013A200409FCEE0', 'Hello XBee 2', 'C2']
['0013A200409FCEE0', 'Hello XBee 2', 'C2']
['0013A200409FCEE0', 'Hello XBee 2', 'C2']
['0013A200409FCEE0', 'Hello XBee 2', 'C2']
['0013A200409FCEE0', 'Hello XBee 2', 'C2']
['0013A200409FCEE0', 'Hello XBee 2', 'C2']
['0013A200409FCEE0', 'Hello XBee 2', 'C2']
['0013A200409FCEE0', 'Hello XBee 2', 'C2']
['0013A200409FCEE0', 'Hello XBee 2', 'C2']
['0013A200409FCEE0', 'Hello XBee 2', 'C2']
['0013A200409FCEE0', 'Hello XBee 2', 'C2']
```

You'll also see messages in the XBee sender application. Sample program output can be seen here:



codes — Python ch07_xbee_sender.py — 80×24

```
                {'default': None, 'len': 2, 'name': 'command'},
                {'default': None, 'len': None, 'name': 'parameter'}],
 'tx': [{'default': '\x10', 'len': 1, 'name': 'id'},
        {'default': '\x00', 'len': 1, 'name': 'frame_id'},
        {'default': None, 'len': 8, 'name': 'dest_addr'},
        {'default': '\xff\xfe', 'len': 2, 'name': 'reserved'},
        {'default': '\x00', 'len': 1, 'name': 'broadcast_radius'},
        {'default': '\x00', 'len': 1, 'name': 'options'},
        {'default': None, 'len': None, 'name': 'data'}]}
send data
send data
send data
send data
send data
send data
send data
send data
send data
send data
send data
send data
send data
send data
```

```
import time
```

```
from xbee import DigiMesh
```

```
import serial


#PORT = 'COM7'

PORT = '/dev/cu.usbserial-A9CNVHXX'

BAUD_RATE = 9600


# Open serial port

ser = serial.Serial(PORT, BAUD_RATE)

# Create API object

xbee = DigiMesh(ser, escaped=True) import pprint

pprint.pprint(xbee.api_commands)

while True:

   try: data = xbee.wait_read_frame() decodedData =
decodeReceivedFrame(data) print(decodedData)

   except KeyboardInterrupt: break

def ByteToHex(byteStr):

   return ".join(["%02X" % ord(x) for x in byteStr]).strip()

def decodeReceivedFrame(data): source_addr =
ByteToHex(data['source_addr']) rf_data = data['data']
```

```
   options = ByteToHex(data['options']) return [source_addr, rf_data,
options]

# xbee with long address

XBEE2_ADDR_LONG = "\x00\x00\x00\x00\x00\x00\xFF\xFF"



while True:

   try: print "send data"

   xbee.tx(frame='0x1', dest_addr=XBEE2_ADDR_LONG,
data='Hello XBee 2') time.sleep(1) except KeyboardInterrupt: break
```

XBee development for Arduino

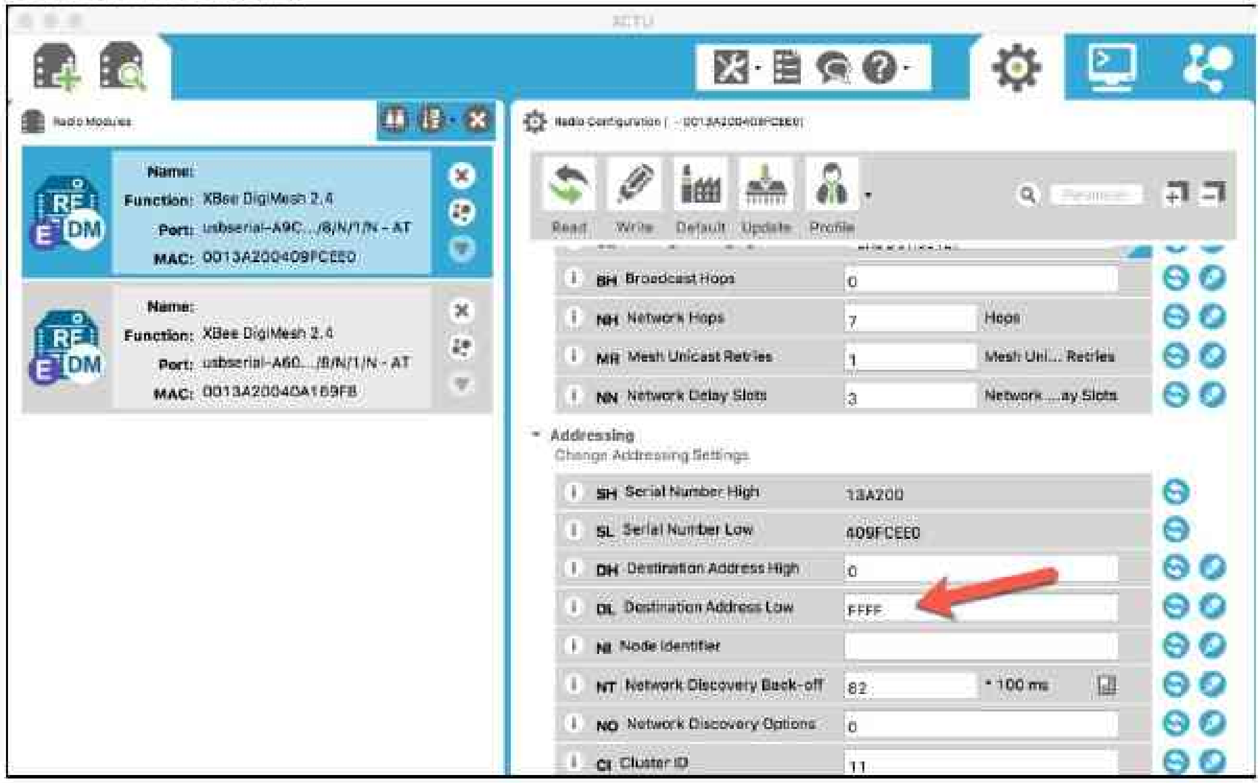
If your robot platform uses Arduino as the MCU model, we can use XBee on the Arduino board. Several Arduino shields for XBee are available as well. Some robot platforms based on Arduino even provide an XBee shield in the board.

In this section, we'll develop an application to communicate between our computer and an XBee module on an Arduino board. We need two XBee modules for this demo.

Configuring the XBee module

We're using a mesh network for communication among XBees. In the previous section, we applied DigiMesh firmware to our XBee modules. In this demo, we'll develop an application to send data from XBee on an Arduino to XBee on a computer.

First, we configure all XBee modules using XCTU. After opening XBee using XCTU, we can start to configure it. Set Destination Address Low (DL) with `FFFF`. You can see it here:

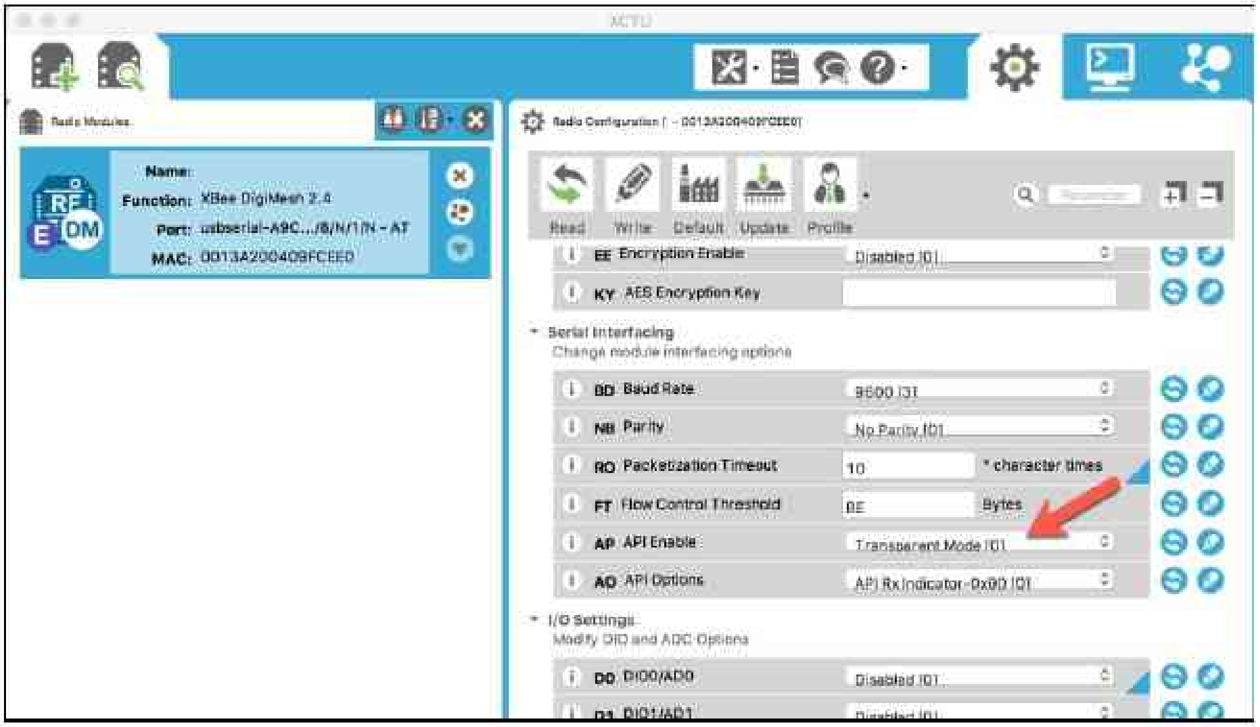

XCTU
Radio Modules
Radio Configuration [- 0013A200409FCEE0]
Read
Write
Default
Update
Profile
Addressing
Change Addressing Settings

Name:
Function: XBee DigiMesh 2.4
Port: usbserial-A9C.../8/N/1/N - AT
MAC: 0013A200409FCEE0

Name:
Function: XBee DigiMesh 2.4
Port: usbserial-A60.../8/N/1/N - AT
MAC: 0013A20040A169FB

BH	Broadcast Hops	0	
NH	Network Hops	7	Hops
MR	Mesh Unicast Retries	1	Mesh Uni... Retries
NN	Network Delay Slots	3	Network ...ay Slots
SH	Serial Number High	13A200	
SL	Serial Number Low	409FCEE0	
DH	Destination Address High	0	
DL	Destination Address Low	FFFF	
NI	Node Identifier		
NT	Network Discovery Back-off	82	* 100 ms
NO	Network Discovery Options	0	
CI	Cluster ID	11	

You also need to set API Enable (AP) as `Transparent Mode [0]`, as shown in the following figure. If finished, you can write all changes to the XBee:



Name:
Function: XBee DigiMesh 2.4
Port: usbserial-A9C.../8/N/1/N - AT
MAC: 0013A200409FCEED

EE	Encryption Enable	Disabled [0]
KY	AES Encryption Key	

Serial Interfacing
Change module interfacing options

BD	Baud Rate	9600 [3]
NB	Parity	No Parity [0]
RO	Packetization Timeout	10 * character times
FT	Flow Control Threshold	BE Bytes
AP	API Enable	Transparent Mode [0]
AO	API Options	API Rx Indicator-0x90 [0]

I/O Settings
Modify DIO and ADC Options

D0	DIO0/AD0	Disabled [0]
D1	DIO1/AD1	Disabled [0]

Implement these steps for both XBee modules.

```
void setup() {

   // start serial Serial.begin(9600); while (!Serial) ;

   // Arduino Leonardo // Serial1.begin(9600); // while (!Serial1) ;

}

void loop() {

   //Serial1.println("hello"); // Leonardo Serial.println("hello");

   delay(1000); }
```

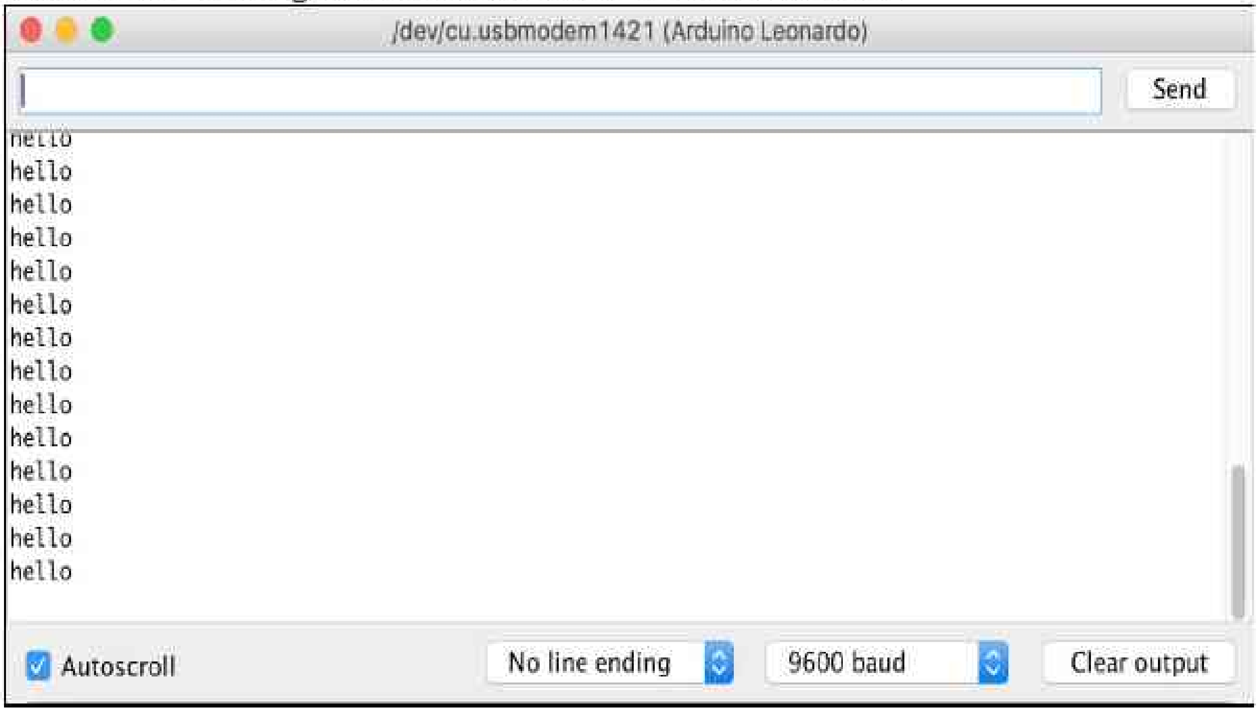
Save this program as `ArduinoXBee`. This program will send the message `hello`. If you use Arduino Leonardo, uncomment the commented section because Arduino Leonardo uses `Serial1` to communicate with XBee.

Testing

Compile and upload the sketch program to your Arduino board. After this, you can open the Serial Monitor tool from the Arduino software.

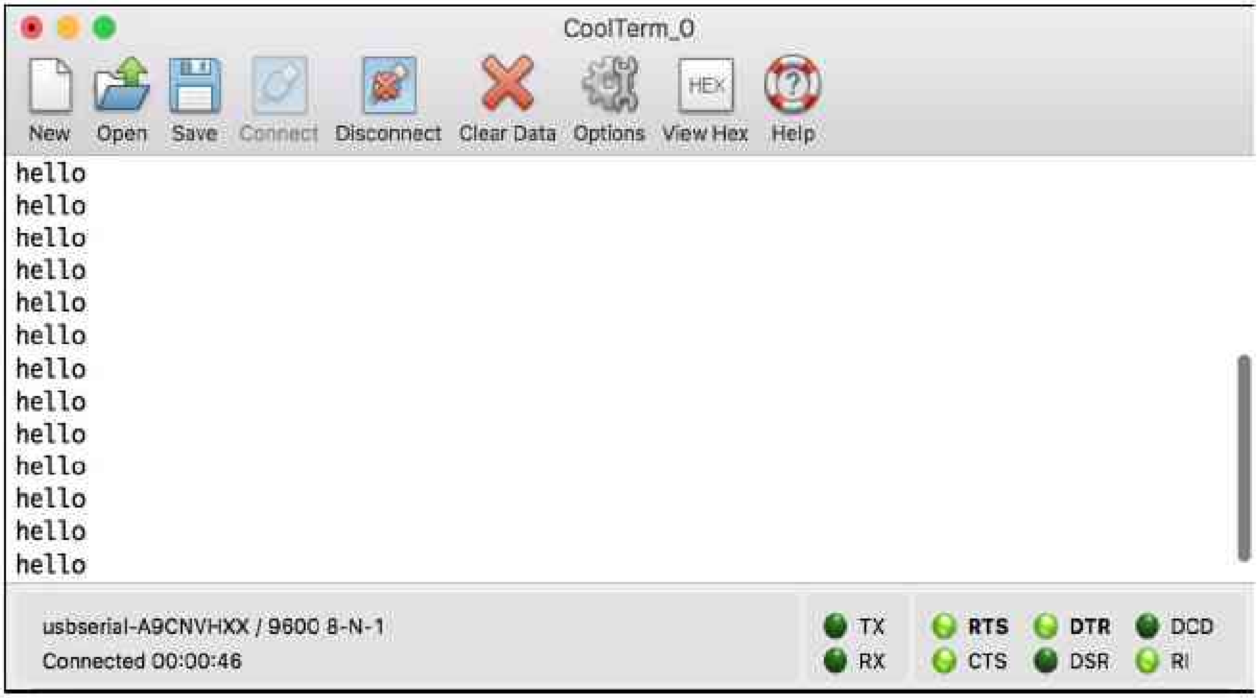
You should also run a serial application such as CoolTerm and open the XBee module to see incoming messages from the Arduino-mounted XBee.

You can see messages from the Arduino in Serial Monitor:



```
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
```

In CoolTerm as well, you can see incoming messages from the Arduino:



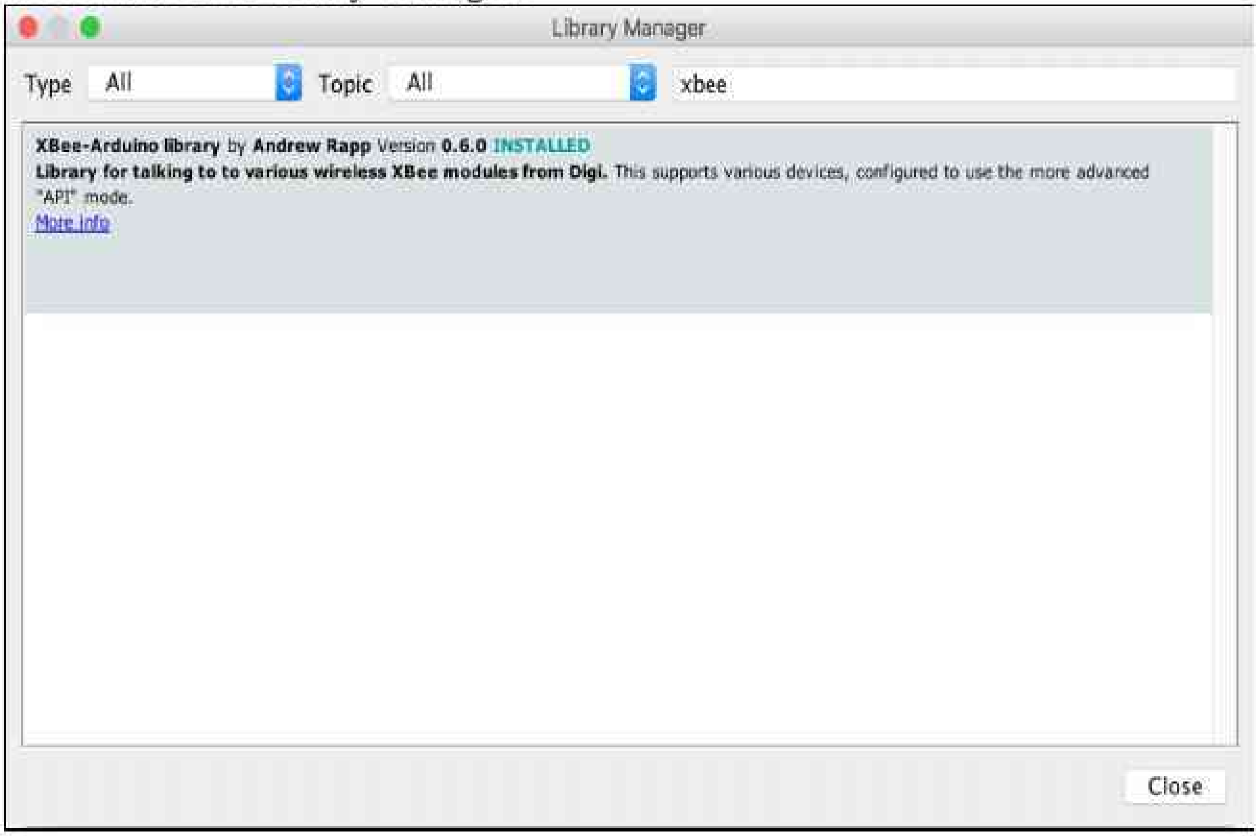
```
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
```

usbserial-A9CNVHXX / 9600 8-N-1
Connected 00:00:46

Working with the XBee library for Arduino

In the previous section, we learned how to communicate with XBee on Arduino through the Serial object. Our XBee works in API Enabled (AP) 0, transparent mode.

You also worked with the XBee in AP 2 mode, so you can control all parameters in XBee, including the destination address of XBee receivers. For sketch programs, we can utilize the Xbee-Arduino library, https://github.com/andrewrapp/xbee-arduino. This library supports XBee Series 1 and Series 2. You also can install it on the Arduino via Library Manager:



XBee-Arduino library by **Andrew Rapp** Version **0.6.0** **INSTALLED**
Library for talking to to various wireless XBee modules from Digi. This supports various devices, configured to use the more advanced "API" mode.

Another library option is the Bee library: https://github.com/kmark/Bee. This library is designed for DigiMesh firmware.

To get started, I recommend you try program samples from these libraries.

Designing a multi-robot cooperation model using swarm intelligence

A multi-robot cooperation model enables some robots to work collectively to achieve a specific purpose. Having multi-robot cooperation is challenging. Several aspects should be considered in order to get an optimized implementation. The objective, hardware, pricing, and algorithm can have an impact on your multi-robot design.

In this section, we will review some key aspects of designing multi-robot cooperation. This is important since developing a robot needs multi-disciplinary skills.

Defining objectives

The first step to developing multi-robot swarm intelligence is to define the objectives. We should state clearly what the goal of the multi-robot implementation is. For instance, we can develop a multi-robot system for soccer games or to find and fight fire.

After defining the objectives, we can continue to gather all the material to achieve them: robot platform, sensors, and algorithms are components that we should have.

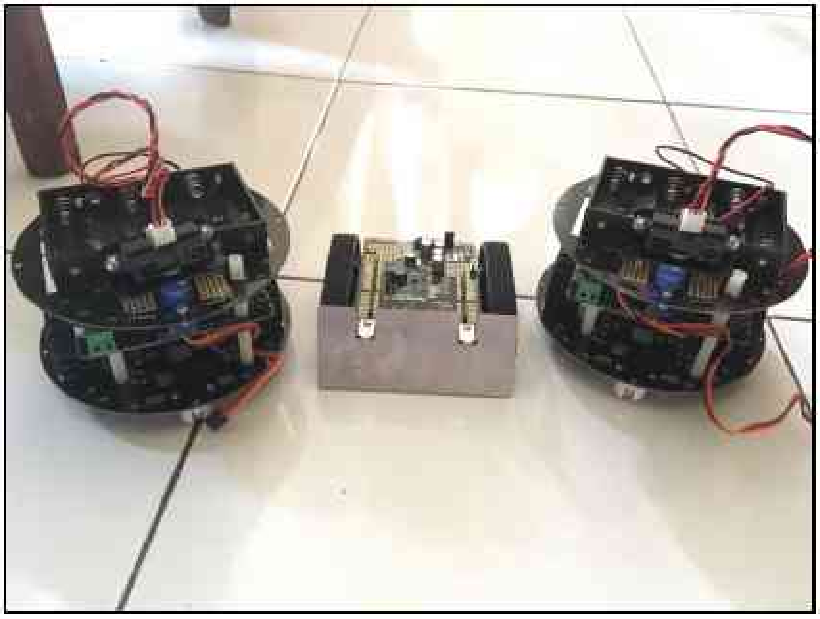
Selecting a robot platform

The robot platform is the MCU model that will be used. There are several MCU platforms that you use for a multi-robot implementation. Arduino, Raspberry Pi, ESP8266, ESP32, TI LaunchPad, and BeagleBone are samples of MCU platforms that can probably be applied for your case.

Sometimes, you may nee to consider the price parameter to decide upon a robot platform. Some researchers and makers make their robot devices with minimum hardware to get optimized functionalities. They also share their hardware and software designs. I recommend you visit Open Robotics, https://www.osrfoundation.org, to explore robot projects that might fit your problem.

Alternatively, you can consider using robot kits. Using a kit means you don't need to solder electronic components. It is ready to use. You can find robot kits in online stores such as Pololu (https://www.pololu.com), SparkFun (https://www.sparkfun.com), DFRobot (https://www.dfrobot.com), and Makeblock (http://www.makeblock.com).

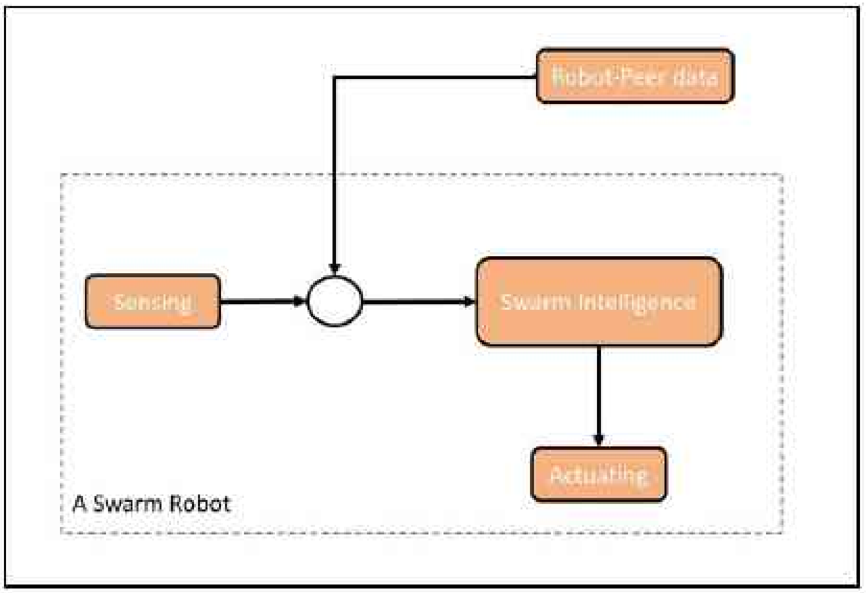
You can see my robots from Pololu and DFRobot here:



Selecting the algorithm for swarm intelligence

The choice of algorithm, especially for swarm intelligence, should be connected to what kind of robot platform is used. We already know that some hardware for robots have computational limitations. Applying complex algorithms to limited computation devices can drain the hardware battery. You must research the best parameters for implementing multi-robot systems.

Implementing swarm intelligence in swarm robots can be described as in the following figure. A swarm robot system will perform sensing to gather its environmental information, including detecting peer robot presence.



By combining inputs from sensors and peers, we can actuate the robots based on the result of our swarm intelligence computation. Actuation can be movement and actions.

Summary

We explored designing multi-robot systems. Key aspects of communication for robots were also reviewed. This chapter can help you get practice in developing several types of robot hardware to build swarm robots. Research should be done to achieve the best performance for a multi-robot system.

Essential Hardware Components

Since this book is in the form of projects, it goes without saying that you are required to have certain pieces of hardware to actually build the projects provided in this book. Here is a list of all the required components along with their Amazon hyperlinks:

Units	Components	Type	Description	Manufacturers	URL
1	Adafruit WICED WiFi Feather	Adafruit WICED WiFi Feather	WiFi Microcontroller Development Board	Adafruit Industries	https:/ goo.gl mRHc Bn
1	MB-102	Solderless breadboard	Solderless breadboard, power supply, and jumper wires	NA	https:/ goo.gl nExwy b
1	JMPR	Dupont Jumper Wires	Haitronic 120pcs 20cm length Jumper Wires/dupont cable Multicolored(10 color) 40pin M to F, 40pin M to M, 40pin F to F for Breadboard / Arduino based / DIY/ raspberry Pi 2 3/Robot Ribbon Cables Kit	NA	https:/ goo.gl HmPD oz
	REED	Winomo Magnetic	Magnetic door		https:/

1	SWITCH	Door Window Contact	switches	Winomo	goo.gl/ 7tP7w
1	Pi3	Raspberry Pi 3 Camera Kit	Raspberry Pi 3 Complete Camera Kit -- Includes Raspberry Pi 3 and 5MP Camera Module	Vilros	https:/ goo.gl/ Dy5t8 N
3	DHT22	DHT22 Temperature Sensor	Gikfun DHT22 AM2302 Temperature And Humidity Sensor for Arduino EK1196_	Gikfun	https:/ goo.gl/ e87cX E
1	OV2640	ArduCam	Arducam Mini Module Camera Shield with OV2640 2 Megapixels Lens for Arduino UNO Mega2560 Board	Arducam	https:/ goo.gl/ uFNnU g