Community Experience Distilled

Building a Home Security System with Arduino

Design, build, and maintain a home security system with Arduino Uno

Jorge R. Castro

[PACKT] open source*
PUBLISHING
community experience distilled

Building a Home Security System with Arduino

Table of Contents

Building a Home Security System with Arduino

Building a Home Security System with Arduino

First published: August 2015

Production reference: 1280815

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham B3 2PB, UK.

ISBN 978-1-78528-060-3

www.packtpub.com

Credits

Author

Jorge R. Castro

Reviewers

Mark de Groot

Aaron Srivastava

Fangzhou Xia

Commissioning Editor

Julian Ursell

Acquisition Editor

Meeta Rajani

Content Development Editor

Sumeet Sawant

Technical Editor

Rohith Rajan

Copy Editor

Charlotte Carneiro

Project Coordinator

Shweta Birwatkar

Proofreader

Safis Editing

Indexer

Tejal Soni

Graphics

Abhinash Sahu

Production Coordinator

Melwyn D’sa

Cover Work

Melwyn D’sa

About the Author

Jorge R.Castro is a young computer engineer who has specialized in new technologies and open source electronics, with vast experience in software design and programming for mobile devices. He is passionate about open source initiatives.

He also dedicates much of his time to computer security and reverse engineering (seeking vulnerabilities in software and analysis of malware).

I also want to acknowledge the help and cooperation of the entire team at Packt Publishing, who have supported me throughout this great project, as well as the organizations responsible for Arduino and Python.

Last but not least, I want to thank you for purchasing this book and taking the first step toward delving into the exciting world of “Maker”.

About the Reviewers

Mark de Groot is an ethical hacker at KPN's (Royal Dutch Telecom) REDteam in Amsterdam, the Netherlands. He has experience in performing penetration tests and security assessments of complex technical environments. He specializes in web penetration tests, infrastructure penetration tests, mobile application security, exploitation development, reverse engineering, fuzzing, network protocol analysis, source code security auditing, intrusion detection, computer forensics, multiplatform development, domotica, and coding of mobile applications.

Mark loves to play CTF (capture the flag) games and has twice finished as runner-up at the World Cyberlympics competition with his team.

Aaron Srivastava is a software engineer at Fujifilm Medical System. He received his bachelors degree in biomedical engineering from North Carolina State University. He has worked on side projects that rely on the Arduino board and other microcontrollers.

Aaron previously has been a reviewer on *Programming Arduino with LABView*, published by *Packt Publishing*. In addition to web development, he enjoys tinkering with other microcontrollers and low-level programming.

Fangzhou Xia is currently a masters student in mechanical engineering (ME) at the Massachusetts Institute of Technology (MIT). He received his bachelor's degree in ME from the University of Michigan, and his bachelor's degree in electrical and computer engineering (ECE) from Shanghai Jiao Tong University.

His areas of interest in mechanical engineering include system control, robotics, product design, and manufacturing automation. His areas of interest include web application development, embedded system implementation, and data acquisition system setup.

www.PacktPub.com

I dedicate this book to Sandra and Isabel, who did not ever stop believing in me.

Also a special dedication to my friends and family, who have always been there for me.

Preface

The Arduino Uno is an open source microcontroller built on a single circuit board that is capable of receiving sensory input from the environment and controlling interactive physical objects. It is also a development environment that allows writing software for the board in the Arduino programming language. It is used for a variety of different purposes and projects, from simple projects such as building a thermostat, to more advanced ones such as robotics, Web servers, seismographs, home security systems, and synthesizers.

This book will demonstrate how Arduino Uno can be used to develop a highly connected home security system by mobilizing a network of sensors, which can feed alerts back to an Arduino Uno when alarms are triggered.

What this book covers

Chapter 1, *Getting Started with a Home Security System*, talks about how traditional home security systems work in principle and gives examples of how connected homes interact with home owners. It also talks about what is needed for the system to be installed and properly maintained.

Chapter 2, *Working with Arduino Uno and Arduino IDE*, deals with what an Arduino Uno is, its history, descriptions of different parts of the Arduino Uno, how they work, and how an Arduino Uno can extend its capabilities with Shields.

Chapter 3, *From Code to the Real World*, teaches us to handle technical documentation fluently, understand the types of signals and their main differences, find the perfect component for our needs, and finally apply this to a real project.

Chapter 4, *Designing Your Own System*, shows you the approaches in designing a home automation system with a slight recap on home automation and Internet of Things (IoT). We'll also see what needs to be taken into account when we design a system. We'll define the priorities of the project, budgets, and robustness.

Chapter 5, *Arduino and Sensors*, shows you how to work with the libraries and what to create, import, and modify to increase the power of your code. Furthermore, we'll integrate more sensors and circuit elements, such as MOSFET and engines, and learn to control them.

Chapter 6, *Documentation and Version Control*, shows you how to document your code, share it, keep track of its status, and maintain a backup of your code. Finally, you'll create a simple graphical application that allows you to remotely control your Arduino Uno.

Chapter 7, *Interaction and Connectivity*, talks about how to create a system for detecting people to increase the capacity of our infrastructure. We have a brief introduction to the key elements of this chapter. Finally, we will conclude with a practical example. New concepts and technologies such as real-time detection of faces and people will be introduced (artificial intelligence).

What you need for this book

To carry out these examples, you will need a ‘starter kit’ (comprising an Arduino Uno board, cables, and resistors; visit https://www.arduino.cc/en/Main/ArduinoStarterKit), a computer, a Raspberry Pi, and an Internet connection.

You will also need to download the Arduino and Python code files from http://www.packtpub.com/support, that are designed to work with the circuit schematics and designs discussed in this book. Make sure you are referring to the code within the code bundle when testing it.

Who this book is for

This book is for novice programmers and hobbyists who want to understand how an Arduino Uno can be used to program a home security system, as well as for those who want to delve deeper into the world of Arduino.

Conventions

In this book, you will find a number of styles of text 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: "We can include other contexts through the use of the `include` directive."

A block of code is set as follows:

```
from time import sleep
import serial
port = serial.Serial('/dev/…', 9600) # "…" Put your serial port
# remember you can know it in the Arduino-IDE
# go to Tools > Serial Port

while True: # Enters a loop in which hears every 0.2 seconds
print port.readline()
sleep(0.2)
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
from time import sleep
import serial
port = serial.Serial('/dev/…', 9600) # "…" Put your serial port
# remember you can know it in the Arduino-IDE
# go to Tools > Serial Port

while True: # Enters a loop in which hears every 0.2 seconds
print port.readline()
sleep(0.2)
```

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

```
sudo apt-get update
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: " At the top of the IDE window, navigate to **Tools** | **Board** and select **Arduino Uno**".

Note

Warnings or important notes appear in a box like this.

Tip

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 may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to <`feedback@packtpub.com`>, and mention the book title via 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 on 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 all Packt books you have purchased 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 e-mailed directly to you.

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 would 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 on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from http://www.packtpub.com/support.

Questions

You can contact us at <questions@packtpub.com> if you are having a problem with any aspect of the book, and we will do our best to address it.

Chapter 1. Getting Started with a Home Security System

In this chapter, we'll cover the following:

- What is home security infrastructure
- How does it work?
- What is needed to install one?
- How to prepare your current home for a security system
- Wired and wireless security systems
- Traditional systems versus the modern home security system

This book comes after the growing trend of hardware devices, which are available around us for a few bucks, and more specifically all talk about the Arduino platform. Today, we find these small boards in such varied and common places like schools, universities, small businesses, or public agencies.

The numerous advantages are well known, such as low cost, open designs, ease of assembly, and ability to make adjustments. That's why almost any project, whether professional or amateur, starts with a small design on a piece of paper and then quickly passes into the physical world using these fabulous boards. We will implement such a technological design to create our domestic security system.

In this chapter, you will be introduced to the world of home security systems. Also, you will be shown the difference between traditional monitoring and home automated systems. You will also learn how they work, the requirements to install them in your home, and how to prepare your environment.

You should delve into this book only after ensuring that you are familiar with the basic concepts of electronics and computers, and later delve into more advanced principles in the subsequent chapters (but no need to worry as each and every point is carefully explained using examples, and will link a lot of information to facilitate understanding). Remember that this book should not be *completely* taken as a step-by-step guide on how to create a foolproof system, but as a tool that will give you the knowledge needed to create your own domestic system.

Note

Safety advisory: Throughout the projects developed here, you should always observe the highest of safety measures, always be attentive of your surroundings, and never work alone if you do not possess adequate knowledge of higher technicalities. Also, try to reduce or eliminate the use of high tension electrical equipment.

What is home security infrastructure?

Surely, if you were asked about the parts of a security system, you would answer the question without any problem. For example, you would talk about surveillance cameras, motion sensors, alarms, and so on, but do you really know all the different pieces that it comprises of? All the connections? How they work? The differences between traditional and current systems? Don't worry. You will have the answers soon.

Well, we have an answer to one of these questions. The security infrastructure comprises of all these components, hardware devices, software elements, and design of their connections, put together. One example is an access point control infrastructure, where you have a card reader connected to a database that checks the permission from the ID of a card (serial number of the card) and associates it with a latch that allows or blocks the access, which is supported by a webcam, to show you the real time condition of any incident.

In this case, if someone tries to break the security measure, the system will trigger an alarm. Then, you can make a decision on what should follow next.

As you can see in this simple example, we have a lot of elements that work together to keep your environment safe. All of these together make a security infrastructure.

How does it work?

We just discussed what *components* are, basically, but what are these elements and how do they work? The first point is to distinguish the types that exist. We can find two kinds of elements in a security system.

The hardware

The physical elements that compose an infrastructure must bear all the technical requirements for software. They can be subdivided into three distinct subgroups. This classification should not be understood as something exclusionary because an element can be found in more than one category, but this separation will help you understand the functions performed in the system:

- **Sensors**: They will be the *senses* of our system, just like the senses a human body has. Their function is to collect information from the environment, transform it into a digital *value*, and deliver it to a component that is designed to control it. The data obtained will be the input for our equipment. This group includes cameras, sound sensors, proximity sensors, motion detection sensors, smoke sensors, infrared sensors, and temperature sensors among others.
- **Actuators**: If the sensors provide us with all the necessary information from our environment, the actuators will be the *muscles* that allow us to perform actions on our surroundings. Once we have made the decision to perform an action, we send a signal and force this element to work. Examples of these are alarms, speakers, locks, and switches.
- **Controller**: This is the brain of the system. A clear example is a microcontroller board, such an as **Arduino** controller, that is able to store a program and run it. It receives the sensory signals, processes them, and then activates and controls the actuator devices and alarms.

The software

As you saw in the last paragraph of the previous point, once we have some input data and want to produce an effect upon its receipt, we need a tool that establishes the rules that set the behavior of our system. This is taken care of by the software resources, which are a set of programs for your system.

More specifically, we will use programming to create our own program and store it on our microcontroller chip. Thus, we get the extracted data that we need (for example, monitor the temperature of a room to detect a fire), process it (calculate the dangers if they exceed a threshold) without our intervention, and implement the necessary measures (trigger alarms and fire extinguishing measures).

At this point, I won't specify a concrete *programming language*. Let's just talk about the software as a *block*, and later delve into different languages and applications (throughout this book, we will use various programming languages, such as **Python**).

At this point of the book, the reader will be able to identify general terms and the elements of a system and understand its functionality. But do you exactly know what you need to begin constructing the system? Where and how to install it? If it is the right place for the installation? You first need to design the system.

The prerequisites for installing a security system

If you are thinking of installing a security system, then most likely it is because there is a crucial need to do so. Hence, the most important thing at this point is to correctly identify this demand. If we fail to do this, we will end up taking the wrong approach and will have a disastrous or maybe incomplete result.

If not provided, and if you have a curious and dexterous mind, you might want to create your own system design just to have fun and learn while you're at it, or modernize your home with specific customized needs. You can take a look at the websites of the leading companies engaged in this sector for ideas.

We see that most have similar systems but the parts that are different are related to the user needs. For example, many use similar or identical electronic components but what differs in much detail is the **Graphical User Interface** (**GUI**), which will be the *face* of the system, and it also depends on the ease and of use and experience required by the end user.) There is no point in having a system if we are not able to handle and understand.

Once you have a design in mind, the next step is to consider the environment in which you want to install the system. Also, consider the material resources at your disposal and the economic cost that you are allowed to take the plunge with.

It may seem simple but the more information you collect, the easier it is to work in successive phases. An illustrative example is to understand the points of light, pipes, and network access that are available to us. Once you have the layout, it then is easy to construct and build our project.

In terms of material throughout this book, we will use one of the most famous Arduino modules—the **Arduino UNO**, revision 3.0. In the next chapter, I'm going to talk more about it. The other ingredients are easily acquired and do not usually have a very high price, yet many of them are created with *free hardware* or *open source hardware* (hardware that are very cheap to source). So if you want, you can build it at home for a very low price compared to the module assembly.

Note

For more information about free hardware, go to: http://en.wikipedia.org/wiki/Open-source_hardware

How to prepare your current home for a security system

Once you have selected the site for the installation, there are a number of measurements that you must take to ensure the security, integrity, and reliability of the system components and even people around it.

Since our system is dependent on electricity, in order to optimize the performance and the costs of use, it is advisable to have a separate point of electrical supply, instead of using batteries, which are expensive and short-lived. In addition, Arduino boards, and many other modules, are equipped with just one connection in which case, you can put in a wall transformer.

If you do not have access to electricity or want to have a backup system (such as a **UPS** (**Uninterruptible Power Supply**), you can use rechargeable **Lithium-Ion** (**Li-ion**) batteries or even implement a system with solar panels. The latter option can be found on numerous examples, such as road signs, farm monitoring systems, weather stations, and so on.

Note

Caution: Whenever batteries are used outdoors, you have to be aware of the battery limitations in extreme temperatures, such as the reduced current carrying capacity is reduced, the current discharge is lower in cold temperatures, and the risk of heat damage, and maybe even explosion in extreme cases.

We also have to be careful with moisture, condensation, and dew because they can spoil our infrastructure.

A good recommendation is to ensure a network connection, not necessarily internet access, but if possible connection to a router, which is a part of an *Intranet* from where you can access the individual components, change settings or access services (for example, a GUI that shows whether an alert has been triggered).

It is very important to note that our site is not prone to extreme radio interference, which may generated by the technology we use constantly. Such devices emit radiation at different frequencies and these can alter the behavior of our system.

On the other hand, if installations are performed outdoors, you should keep them safe from inclement weather, including lightning, and from poor grounding conductors. All this can destroy the circuitry of your system and may put you in danger as well.

Note

Finally, if you have pets at home, be aware of it when you install your sensors and actuators that may have been expensive. This is because they may end up damaged and hardly repairable. Besides, if you do not choose the correct orientation, every time your dog or cat goes through the garden, it may activate an alarm.

Remember that you have to perform maintenance of your devices periodically. I recommend you to set aside a few days on the calendar to remind you and perform basic maintenance.

Wired and wireless security systems.

The last thing that you do is choose how you connect all elements of your system. The market currently offers highly reliable wireless options, despite being easier to deploy, they could raise the overall price of the hardware. An example, which will be discussed in depth in the final chapters, are wireless devices which are able to stand correctly in many places, and allow quick installation (without hardly disturbing the environment with wires).

Also, as mentioned previously, there is the disadvantage of radio interference, as domestic wireless technologies work in very close frequencies. Most popular wireless technologies like Wi-Fi and Bluetooth use 2.4GHz, considering how many devices there are around using that technology, which sometimes coupled with the relative *underpowered* scoping, it often impedes communication.

Note

For more information, I recommend you visit the official website of the agency that regulates the standards of these technologies, IEEE at: http://standards.ieee.org/

The other tendency may be to use the existing wiring in your home. I am referring to the generic wiring in your house that is used to run household electrical items. There are initiatives such as the **X10** technology that connects all the rooms of a building, provided they are on the same electrical phase. It also uses the power lines for signaling and controls. The main disadvantage of this standard of system is the initial economic outlay.

Note

For more information about X10 visit:
http://en.wikipedia.org/wiki/X10_(industry_standard)

Traditional systems versus the modern home security system

Finally, we will present a comparison between traditional systems and current implementations, noting that each still has a specific use, where confronted is the need to reduce model-complexity to a minimum, to increase their reliability, compared to increased usability and richness of information provided.

Initially, after considering the circumstances such as price, the maturity of the technology, and existing infrastructure (connectivity), it is not possible to have systems that can be managed in real-time for a low price.

Today we have smartphones and tablets with sufficient capacity to visualize, receive, and send data collected by our hardware immediately. We can design a system that is implemented on a server created by ourselves, or even use cloud computing for greater processing power, this setup requires zero maintenance.

We should add that the installation, maintenance, and removal can be done by anyone with a basic knowledge of programming and electronics and if someday you relocate, you can easily reuse the most important pieces.

Finally, note that this data access can be done from any part of the world. With an easy-to-use application on your smartphone, you can check that everything is in order at home.

Therefore, we can say that now we have more intelligent systems with more economical and capable prices, which react and measure at a higher speed, offering greater security to our homes and environments.

Summary

We have reached the end of this first introductory chapter, where we detailed the different parts of a security system without going into the exact specifications thereof, in a theoretical way, knowing their elements and functions to perform the same. With all this, we are ready to fully dive into the next chapter. I assure you, it will be intense and full of interesting content.

In the next chapter, we’ll learn about what Arduino is, its history, and descriptions of the different parts on the Arduino UNO. We will also learn how they work, and how an Arduino UNO can be extended with shields.

Chapter 2. Getting Started with Arduino and Arduino IDE

In this chapter, we'll take a look at the following:

- Introduction to Arduino UNO
- Arduino Boards
- Safety precaution
- Arduino IDE
- Hello World
- Python and Arduino

Following the rapid introduction in the previous chapter, we will now embark on the incredible and powerful world of Arduino, discovering its characteristics, programming environment, and functionality. We will even create our first program, but first, let me introduce you to Arduino UNO.

The first ever Arduino was born in Italy in 2005 for students in Ivrea, in order to meet the demand for low cost boards to carry out their projects. Before the existence of this platform, it was hard to come across a device that was truly multipurpose, for under $ 30, with a simple programming interface that is accessible to all, requires no technical knowledge, where the learning curve is really fast, and the programming language is based on **C#** (**C-Sharp**). For instance, back in the day, if we wanted to build a simple system with a light that flashed every *x* amount of time, we had to use a specific chip and manipulate it to flicker often as desired. We would then mount it on a PCB, with some solder and manual connections.

In addition, we must add the fact that since its beginning, it has been promoted as an *open hardware* and has been well received by the community and facilitated toward the existence of the **maker** culture. So, after 10 years of existence and thanks to the Internet, it is highly regarded among the most popular brands in the world.

It is important to note that we can find the same design of a specific board offered by different manufacturers, such as Adafruit, SparkFun, or the Arduino itself.

Arduino is the result of the wonderful open source initiative and also encourages you to download their circuit schematics, improvise its study, to learn, share, and even collaborate on any model. It also enables you to gain knowledge from the community members and the industry, knowing that the license for such a design is usually free.

Note

A quick video on Arduino by Massimo Banzi is available on TED-Talks:

http://youtu.be/UoBUXOOdLXY

Arduino boards

From this point onward, we will work exclusively with the official designs, leaving the characteristics of the other PCB’s for another time. The following outlines how we can classify them according to different characteristics:

- Number of pins for I/O
- Proprietary chip used:
    - ATMEGA family
    - Intel
- Size
- Usage
- Types of connectivity

Arduino Logo

The preceding classification is not exclusive, but it gives a quick overview of the key features that one must seek to find a design that best suits their needs. However, all boards are based on the same principle as explained in this book, and therefore, changing from one to another will not make a big difference. The following are examples of the popular development boards on the market today:

Arduino MEGA

This module has more I/O pins designed to interconnect numerous sensors or actuators, without a multiplexer or anything that allows us to switch signals. Its main use is to be used as a development board as we can control a large amount of hardware without an expansion board. A clear example is a weather station or a 3D printer.

Arduino NANO

This is the smallest of the family. It is such a small module that can be integrated into any device already built, providing it with new properties. It is ideal for adding intelligence to systems, does not require much computing power, and has a minimum consumption (5V) and is lightweight. It has a micro USB port, and all outputs are in the back (which we can connect via cable or welding). Arduino NANO is ideal for final projects, where we can *confine* our microcontroller (such as a drone or a rocket).

Arduino ETHERNET

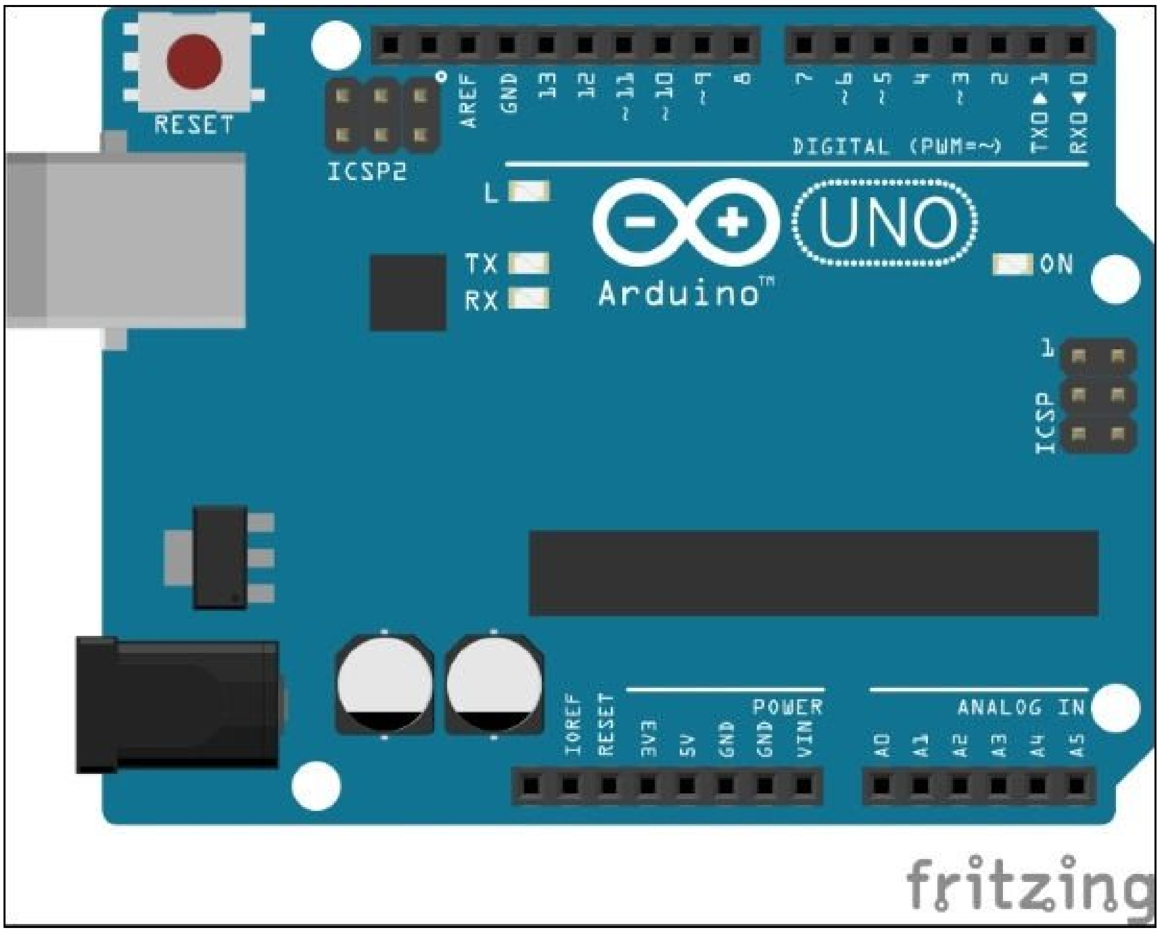
This is very similar to Arduino UNO, but with the great advantage of having an Ethernet port. Now, we should be aware that if we ever use the wired connection, it is better to acquire or build a wireless module.

Intel GALILEO

One of the latest to join the boards with a family of Intel processors is the Galileo, along with the Edison. Galileo has an Ethernet port, two micro USB (bidirectional) ports, and a multiple I/0 that brings the possibility of creating a cluster of these boards and making them work as a single unit.

Arduino UNO

This is the most versatile and widespread of all boards, with an average size allowing it to be placed in almost any space. At first glance it may seem simple, but it has a large number of expansion modules (such as Wi-Fi, Bluetooth, NFC, and I/O expansion module boards), which allow us to have what we need when we need it (just in time). This makes it a very versatile microcontroller. It has greater support from developers than any other board on the market today.



Arduino UNO (Fritzing tool)

It has 13 digital ports, 6 analog, a 5V, and a 3.3 V port. Besides, the mountable ATMEGA328 chip is replaceable, which is an important feature for possible future modifications or for repair.

Tip

Hacker trick

In some cases, you can unplug the microcontroller chip and use your Arduino board as a serial port adapter, a technique that is not covered in detail in this book. (Caution: by doing this, you may be risking the integrity of the device).

With the chip removed, we will be using pins 0 (RX) and 1 (TX).

Note

For more information about the specification, features or to purchase the various kinds of boards, visit the following links:

- http://www.arduino.cc/en/pmwiki.php?n=Main/Boards
- http://store.arduino.cc/
- https://www.sparkfun.com/
- https://www.adafruit.com/

Safety precautions

Children can participate in building the projects in this book, but it is essential that its always done under the supervision of a responsible adult to observe security measures. Although our module operates at a very small voltage 3.3V to 5V, carelessness or improper use of certain techniques can endanger the user. Therefore, among others, the following precautions are recommended:

- Always observe a clean and uncluttered workspace.
- When working with electricity or heat sources, you should keep away from flammable products. A garage full of paints, solvents, or chemicals is not the right place for welding.
- Arduino is perfect for kids, there is no need for solder wires, but it consists of small parts, so handle them with care.
- If you work with high voltage, never handle the modules or components without turning off the power supply before.
- It is highly recommended that you protect your project material from harsh weather conditions. Test your equipped project for a prolonged period before installing it at a final location (this is to ensure the stable operation of the components and increase security).
- If at all you are stuck and do not know what you are doing, or have questions, there are numerous ways to solve problems. You can try to simplify the problem by looking for a solution on the Internet community (there are numerous forums for makers), or by performing a schematic assembly and carefully analyzing it. If you know that something is as Devéria in a facility, you can try to identify the damaged component with a multimeter.

We are all humans and can make mistakes, and this also includes people who voluntarily upload their designs to the web. It is always advised that you contact the person whose design you have selected, to find out more about the design. This will help you to learn more, have a good time, and also make sure that you understand why and how things work.

Arduino IDE

Let’s begin with the installation process.

Installation

Once we have refreshed some concepts, we will move on to the practical part. For this book, you will be using an Arduino UNO, a USB Type A male connector to USB female Type B connector wire and a PC or alternatively a Raspberry Pi.

Then we will cover the steps for specifying various facilities by operating system. Those who sipongan executable files just mentioned superficially, and those requiring certain complexity, will be treated in more detail. (For other distributions or if you have trouble installing you can check out the online guide).

To download the Arduino IDE, visit: http://arduino.cc/en/Main/Software

Installing IDE on Microsoft Windows

To install the Arduino IDE on Windows OS, you may need administrative rights. Once you are sure you have the necessary permissions, follow the given steps:

1. Download and extract the ZIP file on your desktop
2. Run the executable setup file (`.exe` file) by double-clicking on it
3. Once the installation is complete, you can connect the module to the PC

Installing IDE on Mac OS

For installing Arduino IDE on a Mac OS system, you may need administrative rights. Once you have the permission, follow the given steps:

1. Download and extract the ZIP file on your desktop
2. Run the `.dmg` setup file and follow the on screen instructions
3. Once the installation is complete, you can connect the module to your Mac

GNU-Linux (Debian-Ubuntu)

To install the Arduino IDE on a Linux system, follow the given steps:

1. Open a terminal and run the following commands after you make sure that you have **superuser** permissions (you know you have superuser access, when commands typed at start have a dollar sign (`$`) before them:

   ```
   George:~$ sudo apt-get update             //Updating repository
   George:~$ sudo apt-get install arduino   //Downloading & install
   George:~$ echo $?                        //Test
   ```

2. After the last command if you get a `0` character printed on-screen, then it indicates that the installation is completed successfully. If not, repeat the preceding steps carefully. If the error persists, consult the official guide on the website of Arduino.
3. If you are using some other Linux distribution that has no **Advanced Package Tool** (**APT**) to download a file from the official `.tgz` repository (Arduino website), perform the following steps:

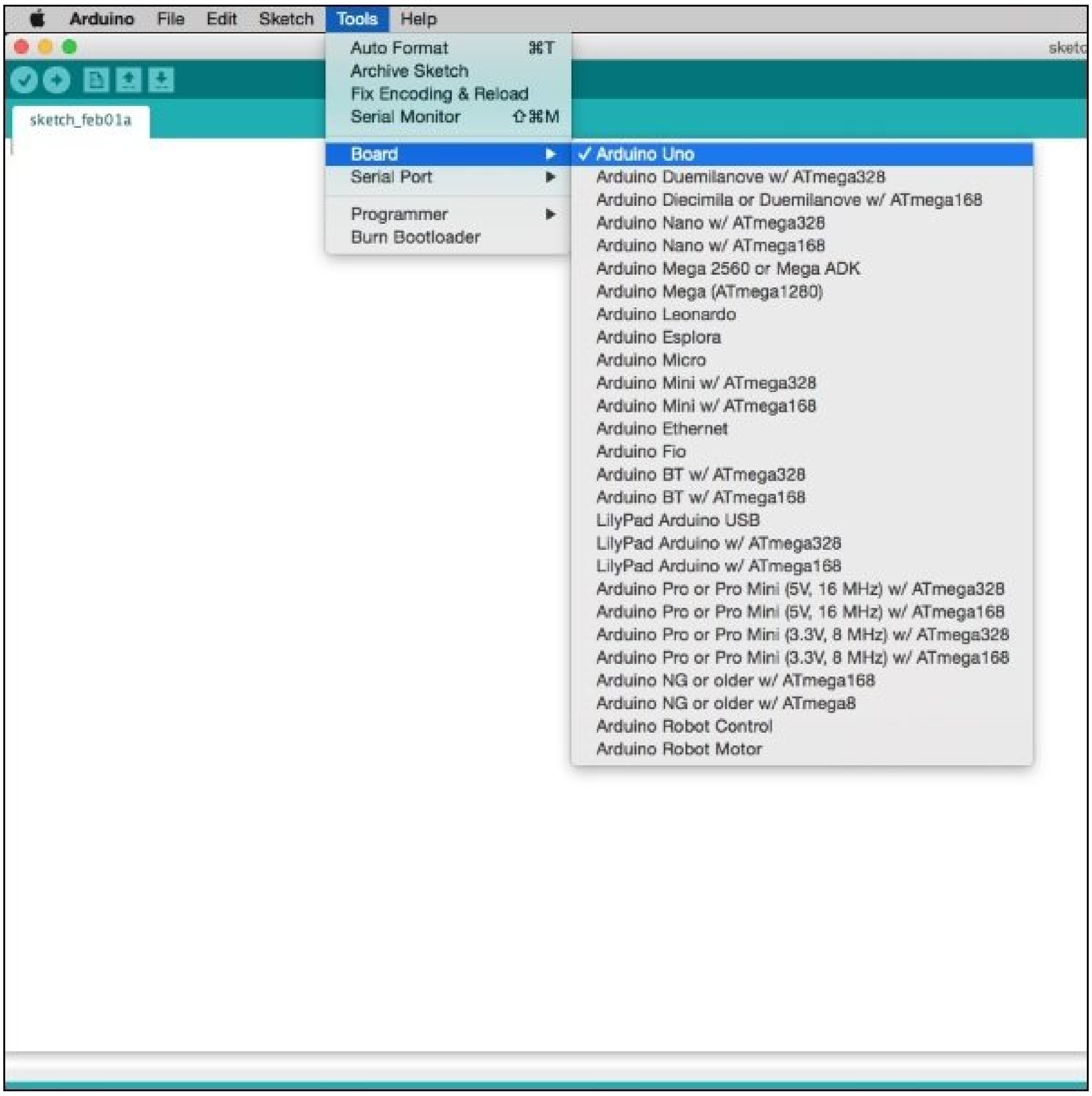
   ```
   George:~$ tar xvzf (filename.tgz)        //untar file
   ```

```
George:~$ cd Arduino-1.X.Y…    //Changing directory to
George:~$ ./arduino           //Test
```

Working with Arduino IDE

Once you have successfully installed the Arduino IDE, let's proceed to learn its components and see how to configure it to your Arduino UNO, which must already be connected to your computer. Follow the given steps to configure the IDE:

1. At the top of the IDE window, navigate to **Tools** | **Board** and select **Arduino UNO**.


Arduino
File
Edit
Sketch
Tools
Help
sketch_feb01a
Auto Format ⌘T
Archive Sketch
Fix Encoding & Reload
Serial Monitor ⇧⌘M
Board
Serial Port
Programmer
Burn Bootloader
✓ Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora
Arduino Micro
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino USB
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8
Arduino Robot Control
Arduino Robot Motor

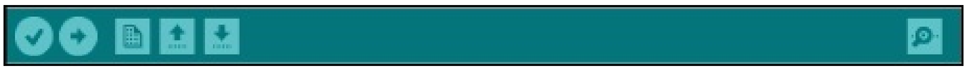
Arduino IDE- Selecting Arduino UNO

2. Then, navigate to **Tools** | **Serial Port** to select a serial port to connect with Arduino UNO. Try selecting a different port number and if one does not work, it may conflict with a physical or virtual device connected to your computer.

With this, we have set up our software environment to start working on the project. Easy right? For others who have experience with other microcontrollers, will realize how easy it is to work the platform and how simplified the environment is.

The following are the different on-screen details of the friendly environment:

- At the top bar, you will see a small green bar, which has a few icons, such as a check, and a magnifying glass at the extreme right-hand side of the bar. The following are details on these icons:
    - **Verify**: This will let you check the code before loading into the chip. It compiles and keeps a transformation of our code in a language that the machine understands
    - **Upload**: This lets you burn a code on the Arduino UNO
    - **New**: With this option, you can open a blank project file to write a new fresh code
    - **Open**: This lets you open one of your saved code files stored on your computer
    - **Save**: This lets you save your code file.
    - **Serial Monitor**: This is one of the most used resources, and it enables us to communicate with our project. Basically, it is an interface that provides the ability to view the signals that is sent to and received by Arduino UNO.



Arduino IDE- Details

- The white canvas is where you will type in the code for our project, we will introduce the code later that will be verified, compiled, and sent to the Arduino UNO.
- Finally, the bottom of the screen will have the console (black box): it’s a tool that will let us know if our code has errors in it, and it will show them here.

Hello World

It is time to begin understanding the essential counterpart of your device—the software/code. Arduino UNO can be programmed in multiple languages, but for the neophyte or if we seek a quick and elegant solution, we can use the language that was used by the designer of a particular project (which is a variant of processing).

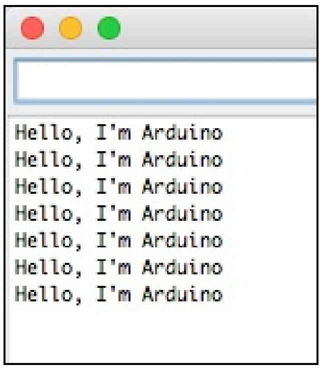
The structure of a program is as follows- it serves as a *template* for all of our future programs:

```
// This line is a comment and IDE can't read it
// WE MUST USE COMMENT to support the code and readability

#include NAMELIBRARY         //change NAMELIBRARY for your library

void setup(){                //Will run once, and use it to
                             //prepare I/0 pins of Arduino

pinMode(10, OUTPUT);         //assign pin 10 as an output
pinMode(11, INPUT);          //assign pin 11 as an input

}

void loop(){                // Is executed just after the "setup"
                            // and repeated countless times
}
```

With this, we already have enough knowledge to create a small program. If you wish, you can study other code examples that come bundled with the IDE. Go to **File** | **Examples**, navigate to `C:\Program Files (x86)\Arduino\examples\`, and open any of the projects to view and experiment with the code (provided you have the necessary hardware required for the example). The following is a basic example of code used for the Arduino UNO:

```
//My First Arduino Program

void setup()
{
  Serial.begin(9600);                  //Open Serial Port
}

void loop()
{

  delay(1000);                          //Wait 1sec =1000 millisec

 Serial.print("Hello, I'm Arduino\n");  // Print "Hello..
  // "\n" is like "ENTER KEY"

}
```

The preceding example performs the simple task of opening the serial port (only once) and waits for 1000 milliseconds. It then prints the desired message.

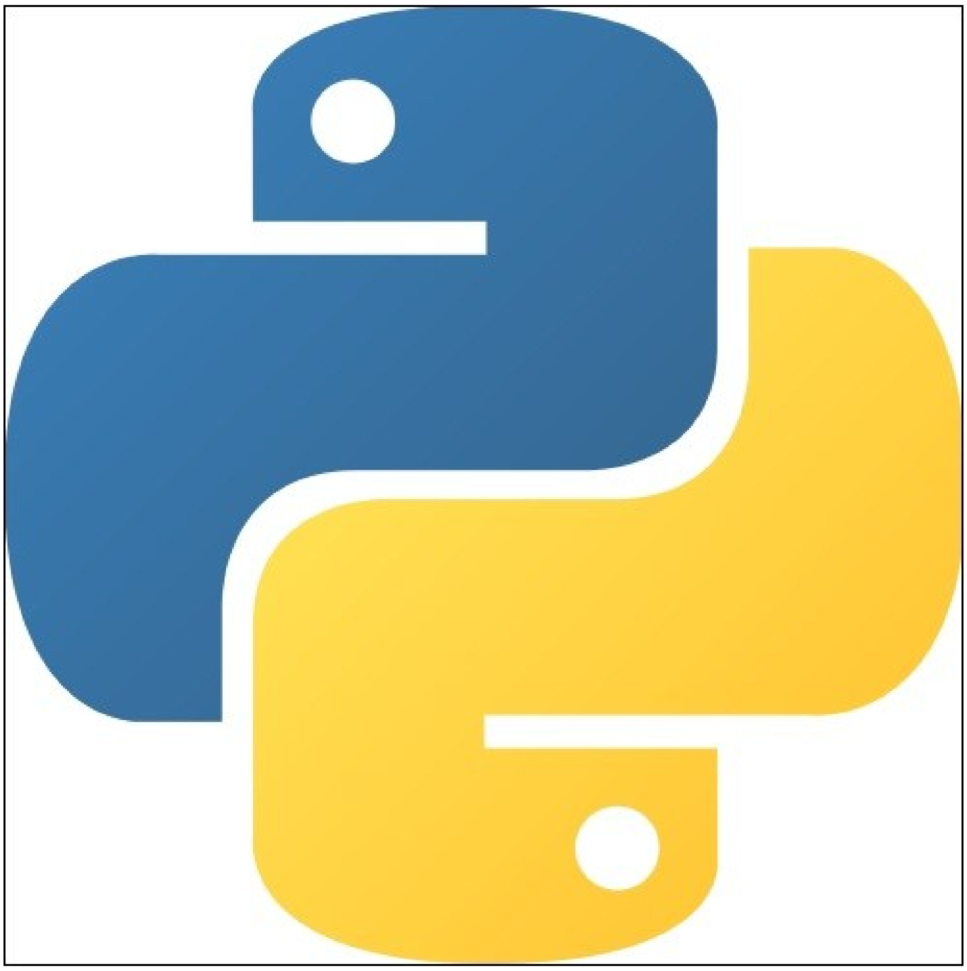


```
Hello, I'm Arduino
Hello, I'm Arduino
Hello, I'm Arduino
Hello, I'm Arduino
Hello, I'm Arduino
Hello, I'm Arduino
Hello, I'm Arduino
```

Arduino Serial Port – Output

Python and Arduino

Once we understand how the serial port works, we will take one more step. We will use **Python** to communicate with Arduino UNO, without using the serial monitor.



Python-logo

As a quick refresher, Python is an interpreted language, designed to increase the readability of the code, to be modular, flexible, and multipurpose. With Python, we can make mathematical calculations, complex equations, create graphic interfaces, and even sounds and images. It's really hard to give a brief overview of this fantastic programming language, and this is why I urge you to consult the numerous reference books on Python.

For newcomers to this language, as a start, visit the following link to get the basic idea of Python: https://www.python.org/

To proceed with this section, you need to have installed Python 2.7 and be familiar with the console or with any graphic environment program. If this is the first time you are hearing of all this, do not worry. The preceding link should give simple tutorials and guides to enable you to learn quickly and be able to continue.

PySerial

Start downloading the compressed file official site: https://pypi.python.org/pypi/pyserial (`pyserial-2.7.tar.gz` for Linux, or `pyserial-2.7.win32.exe` for Windows.)

To install PySerial follow the given steps:

- On Windows, execute the `.exe` file and follow the onscreen instruction to complete installation.
- On Mac/Linux, you need to untar the appropriate downloaded file. Use the following command to untar the file from terminal:

  ```
  tar xfvz /Downloads/pyserial-2.6.tar.gz
  ```

- Change directory and install using the following commands:

  ```
  cd pyserial-2.6
  sudo python setup.py install
  ```

Arduino Code

Now, we re-use the Arduino code that prints a periodic message in from the serial port.

Python Code

We will create a Python code that receives a message from the same serial port. The following is the code:

```
# This is a comment
from time import sleep
import serial
port = serial.Serial('/dev/…', 9600) # "…" Put your serial port
# remember you can know it in the Arduino-IDE
# go to Tools > Serial Port

while True: # Enters a loop in which hears every 0.2 seconds
print port.readline()
sleep(0.2)
```

You can create it with your favorite text editor and save it with the name `SerialPython.py`.

Finally, and without opening the **Serial Monitor** in the Arduino IDE, return the program onboard. After that, it will invoke the Python program. For example, open a shell and assume that the `SerialPython.py` file is in the current directory, type:

```
$ python SerialPython.py
```

The output will look like this:

```
HELLO SERIAL
HELLO SERIAL
HELLO SERIAL
```

Summary

We have reached the end of this intense chapter, where we ventured into the world of Arduino, discovered the different components, how to manage safety precautions, and we finally installed and created our first program.

This chapter is very essential to later chapters in this book, so make sure you have thoroughly understood the concepts explained in this chapter. Once you are ready, move on to the next chapter, where you will dive fully into the world of sensors.

Chapter 3. From Code to the Real World

Once you are clear about the important points in the previous chapters, we can expand our knowledge. After completing this chapter, you will understand things much like reading a technical specification sheet, which are the ports and how they work, the necessary elements for a *proof of concept*. Finally, we will extend the capabilities of our script in Python and rely on NFC technology.

In this chapter, you will cover the following points:

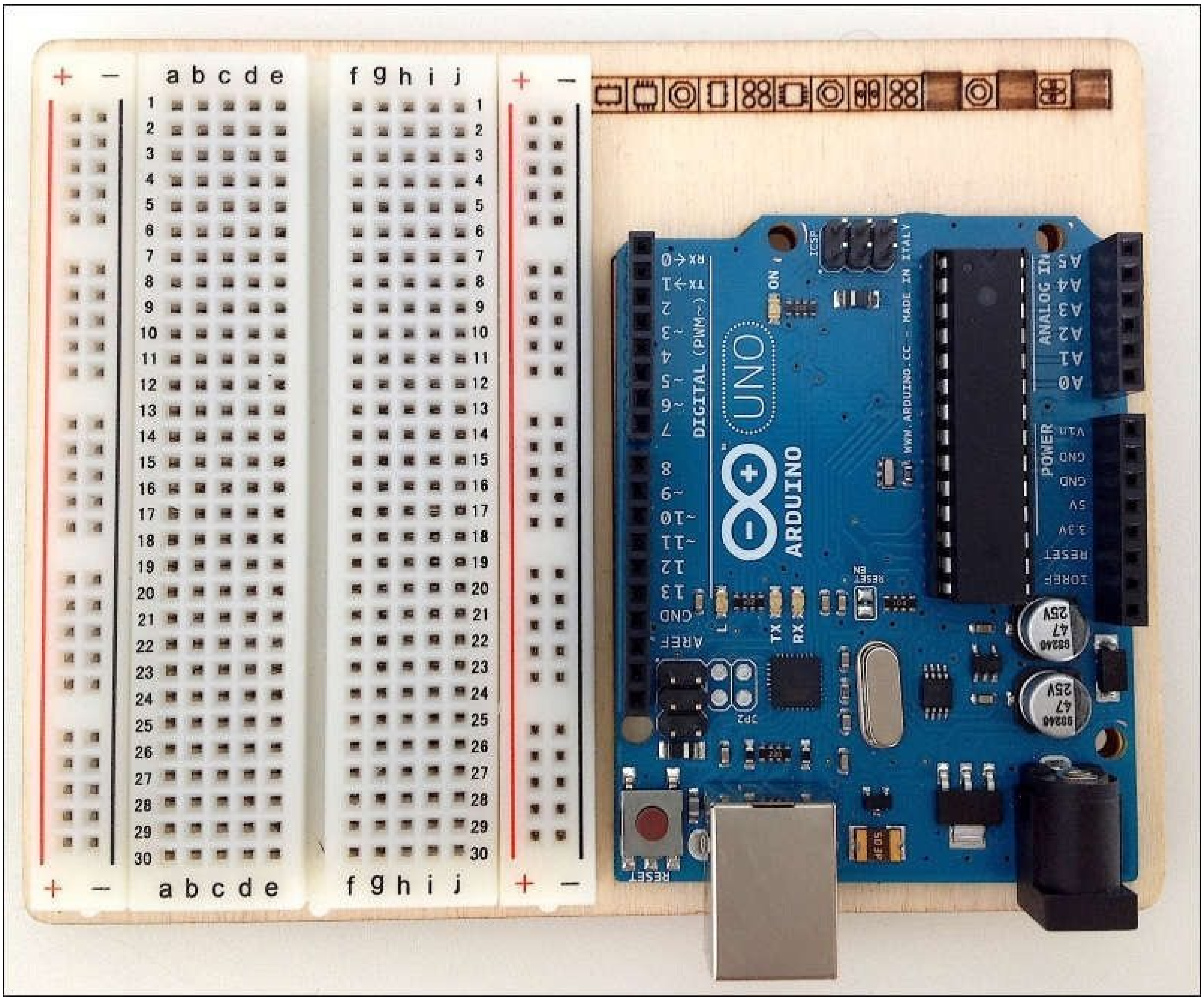
- ProtoBoards and wiring
- Signals (digital and analog)
- Ports
- Datasheets

ProtoBoards and wiring

There are many ways of working on and testing electrical components, one of the ways developers use is using **ProtoBoards** (also known as **breadboards**). Breadboards help eliminate the need to use soldering while testing out components and prototype circuit schematics, it lets us reuse components, enables rapid development, and allows us to correct mistakes and even improve the circuit design. It is a rectangle case composed of internally interconnected rows and columns; it is possible to couple several together. The holes on the face of the board are designed to in way that you can *insert* the pins of components to mount it.

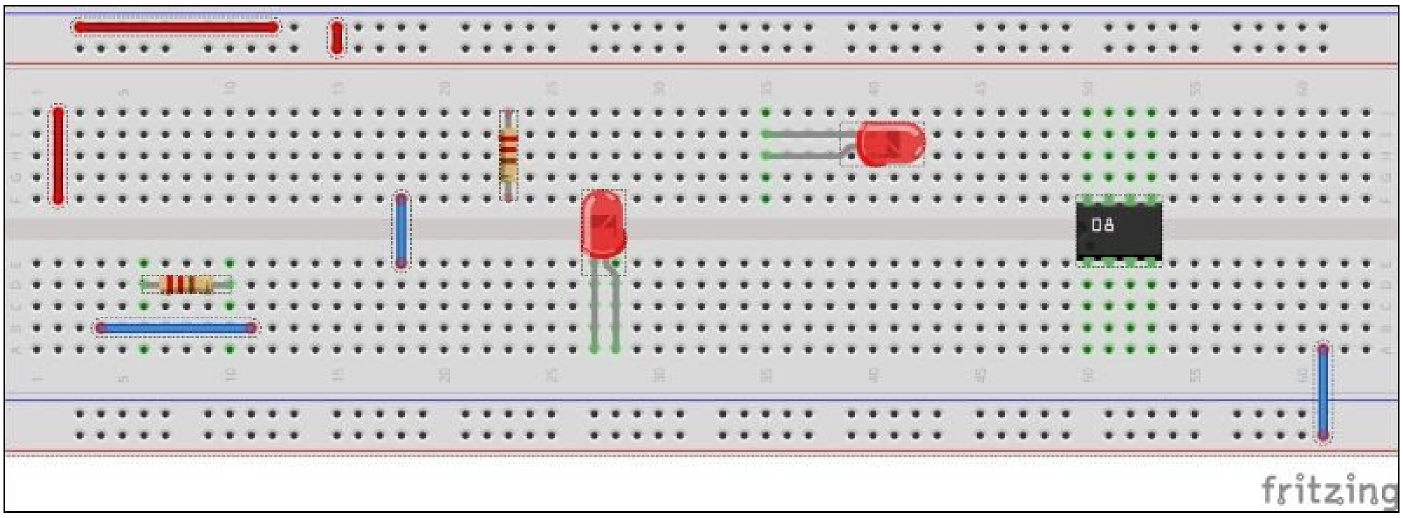
ProtoBoards may be the intermediate step before making our final PCB design, helping eliminate errors and reduce the associated cost. For more information visit http://en.wikipedia.org/wiki/Breadboard.

We know have to know more about the wrong wiring techniques when using a breadboard. There are rules that have always been respected, and if not followed, it might lead to shorting out elements that can irreversibly damage our circuit.



The ProtoBoard is shown in the preceding image; we can see there are mainly two areas - the vertical columns (lines that begin with the + and - polarity sign), and the central rows (with a coordinate system of letters and numbers). Both + and - usually connect to the electrical supply of our circuit and to the ground. It is advised not to connect components directly in the vertical columns, instead to use a wire to connect to the central rows of the breadboard. The central row number corresponds to a unique *track*. If we connected one of the leads of a resistor in the first pin of the first row then other lead should be connected to any other pin in another row, and not in the same row.

There is an imaginary axis that divides the breadboard vertically in symmetrical halves, which implies that they are electrically independent. We will use this feature to use certain **Integrated Circuits** (**IC**), and they will be mounted astride the division.



Protoboard image obtained using Fritzing.

Carefully note that in the preceding picture, the top half of the breadboard shows the components mounted incorrectly, while the lower half shows the components mounted the correct way. A chip is set correctly between the two areas, this usage is common for IC’s.

Note

Fritzing is a tool that helps us create a quick circuit design of our project. Visit http://fritzing.org/download/ for more details on Fritzing.

Analog and digital ports

Now that we know about the correct usage of a breadboard, we now have another very important concept - ports. Our board will be composed of various inputs and outputs, the number of which varies with the kind of model we have.

But what are ports? The Arduino UNO board has ports on its sides. They are connections that allow it to interact with sensors, actuators, and other devices (even with another Arduino board). The board also has ports that support digital and analog signals. The advantage is that these ports are bidirectional, and the programmers are able to define the behavior of these ports.

In the following code, the first part shows that we set the status of the ports that we are going to use. This is the `setup` function:

```
//CODE
void setup(){                      // Will run once, and use it to
                                   // "prepare" I/0 pins of Arduino

pinMode(10, OUTPUT);               // put the pin 10 as an output
pinMode(11, INPUT);                // put the pin 11 as an input
}
```

Lets take a look at what the digital and analog ports are on the Arduino UNO.

Analog ports

Analog signals are signals that continuously vary with time These can be explained as voltages that have continuously varying *intermediate* values. For example, it may vary from 2V to 3.3V to 3.0V, or 3.333V, which means that the voltages varied progressively with time, and are all different values from each other; the figures between the two values are *infinite* (theoretical value). This is an interesting property, and that is what we want. For example, if we want to measure the temperature of a room, the temperature measure takes values with decimals, need an analog signal. Otherwise, we will lose data, for example, in decimal values since we are not able to store infinite decimal numbers we perform a mathematical *rounding* (truncating the number). There is a process called discretization, and it is used to convert analog signals to digital.

Note

In reality, in the world of microcontrollers, the difference between two values is not infinite. The Arduino UNO's ports have a range of values between 0 and 1023 to *describe* an analog input signal. Certain ports, marked as PWM or by a ~, can create output signals that vary between values 0 and 255.

Digital ports

A digital signal consists of just two values - 0s and 1s. Many electronic devices internally have a *range* currently established, where voltage values from 3.5 to 5V are considered as a logic 1, and voltages from 0 to 2.5V are considered as a logic 0.

To better understand this point, let's see an example of a button that triggers an alarm. The only useful cases for an alarm are when the button is pressed to ring the alarm and when it's not pressed; there are only two states observed. So unlike the case of the temperature sensor, where we can have a range of *linear* values, here only two values exist. Logically, we use these properties for different purposes.

Note

The Arduino IDE has some very illustrative examples. You can load them onto your board to study the concepts of analog and digital signals by navigating to **File** | **Examples** | **Basic** | **Blink**.

Sensors

In Chapter 1, *Getting Started with a Home Security System*, we discussed the different categories of hardware, and now we will expand on one of them – sensors.

There is a wide range of sensors, and without pretending to be an accurate guide of all possible sensors, we will try to establish some small differences.

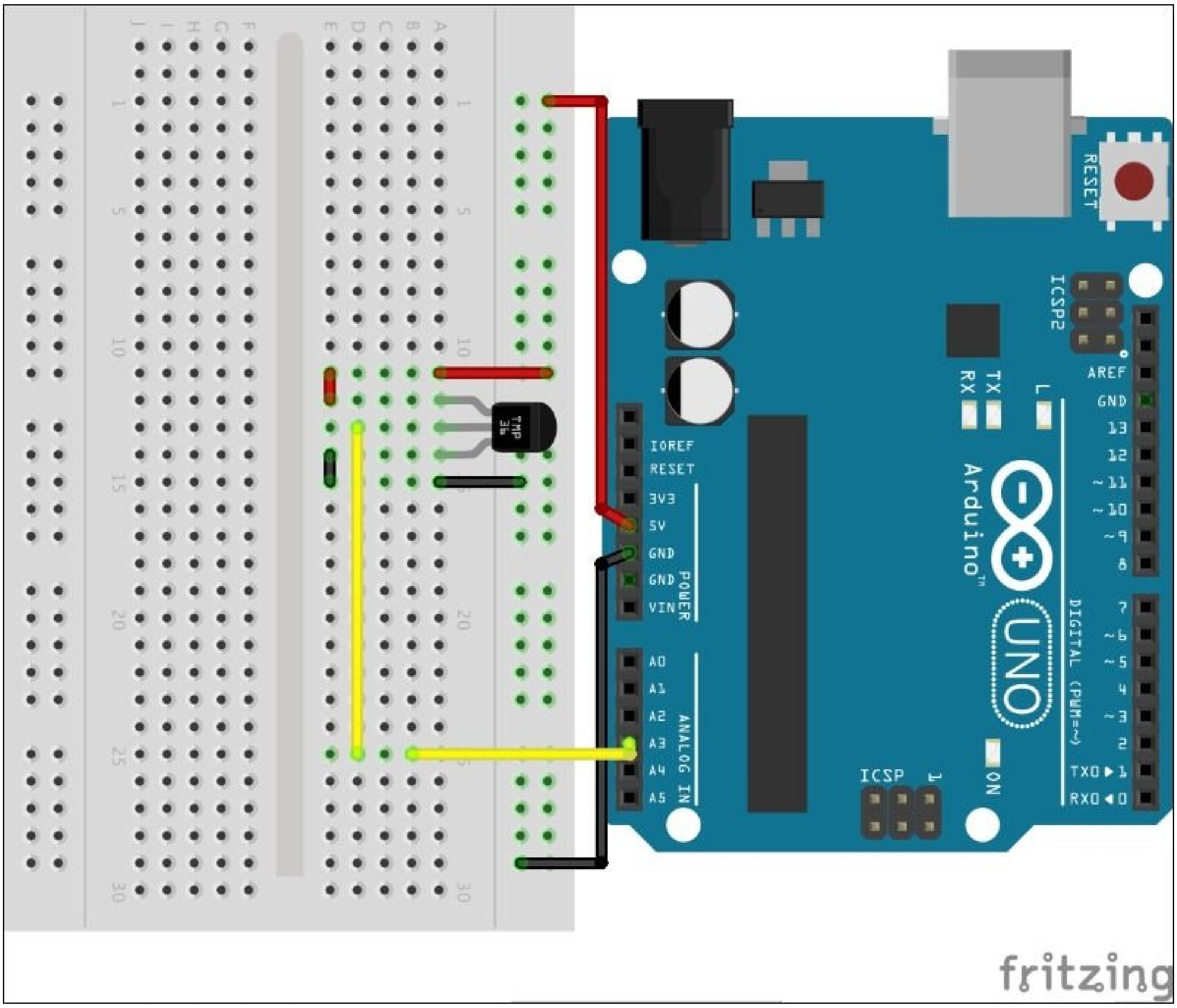
All physical properties or sets of them has an associated sensor, so we can measure the force (presence of people walking on the floor), movement (physical displacement even in the absence of light), smoke, gas, pictures (yes, the cameras also are sensors) noise (sound recording), antennas (radio waves, WiFi, and NFC), and an incredible list that would need a whole book to explain all its fantastic properties.

It is also interesting to note that the sensors can also be specific and concrete; however, they only measure very accurate or nonspecific properties, being able to perceive a set of values but more accurately. If you go to an electronics store or a sales portal buy a humidity sensor (and generally any other electronic item), you will see a sensitive price range. In general, expensive sensors indicate that they are more *reliable* than their cheaper counterparts, the price also indicates the kinds of conditions that it can work in, and also the duration of operation. Expensive sensors can last more than the cheaper variants.

Note

When we started looking for a component, we looked to the datasheets of a particular component (the next point will explain what this document is), you will not be surprised to find two very similar models mentioned in the datasheets. It contains operating characteristics and different prices. Some components are professionally deployed (for instance in the technological and military sectors), while others are designed for general consumer electronics.

Here, we will focus on those with average quality and foremost an economic price. We proceed with an example that will serve as a liaison with the following paragraph. To do this, we will use a temperature sensor (TMP-36) and an analog port on the Arduino UNO. The following image shows the circuit schematic:



Our design - designed using Fritzing

The following is the code for the preceding circuit schematic:

```
//CODE
//#############################
//Author : Jorge Reyes Castro
//Arduino Home Security Book
//#############################
// A3 <- Input
// Sensor TMP36
//#############################
//Global Variable

int outPut = 0;
float outPutToVol = 0.0;
float volToTemp = 0.0;

void setup(){
 Serial.begin(9600);      // Start the SerialPort 9600 bauds
}

void loop(){
 outPut = analogRead(A3);

// Take the value from the A3 incoming port

 outPutToVol = 5.0 *(outPut/1024.0);

 // This is calculated with the values

 volToTemp = 100.0 *(outPutToVol - 0.5);

 // obtained from the datasheet

Serial.print("_____________________\n");
 mprint("Output of the sensor   ->", outPut);
 mprint("Conversion to voltage   ->", outPutToVol);
 mprint("Conversion to temperature ->", volToTemp);
 delay(1000);           // Wait 1 Sec.

}
// Create our print function
// smarter than repeat the code and it will be reusable

void mprint(char* text, double value){

// receives a pointer to the text and a numeric value of type double

 Serial.print(text);
 Serial.print("\t");           // tabulation
 Serial.print(value);
 Serial.print("\n");           // new line
}
```

Now, open a serial port from the IDE or just execute the previous Python script. We can see the output from the Arduino. Enter the following command to run the code as python script:

```
$ python SerialPython.py
```

The output will look like this:

```
_____________________
Output of the sensor          145.00
Conversion to voltage           0.71
Conversion to temperature      20.80
_____________________
```

By using Python script, which is not simple, we managed to extract some data from a sensor after the signal is processed by the Arduino UNO. It is then extracted and read by the serial interface (script in Python) from the Arduino UNO. At this point, we will be able to do what we want with the retrieved data, we can store them in a database, represent them in a function, create a HTML document, and more.

As you may have noticed we make a mathematical calculation. However, from where do we get this data? The answer is in the data sheet.

Component datasheets

Whenever we need to work with or handle an electrical component, we have to study their properties. For this, there are official documents, of variable length, in which the manufacturer carefully describes the characteristics of that component.

First of all, we, have to identify that a component can either use the unique ID or model name that it was given when it was manufactured.

Note

TMP 36GZ

We have very carefully studied the component surface and thus extracted the model. Then, using an Internet browser, we can quickly find the datasheet. Go to http://www.google.com and search for: `TMP36GZ + datasheet`, or visit http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Temp/TMP35_36_37.pdf to get the datasheet.

Once you have the datasheet, you can see that it has various components which look similar, but with very different presentations (this can be crucial for a project, as an error in the choice of *encapsulated* (coating/presentation) can lead to a bitter surprise). Just as an example, if we confuse the presentation of our circuit, we might end up buying components used in cellphones, and those can hardly be soldered by a standard user as they smaller than your pinkie fingernail). Therefore, an important property to which you must pay attention is the coating/presentation of the components.

In the initial pages of the datasheet you see that it shows you the physical aspects of the component. It then show you the technical characteristics, such as the working current, the output voltage, and other characteristics which we must study carefully in order to know whether they will be consistent with our setup. In this case, we are working with a range of input voltage (V) of between 2.7 V to 5.5 V is perfect. Our Arduino has an output of 5V. Furthermore, the temperature range seems appropriate. We do not wish to measure anything below 0V and 5V. Let’s study the optimal behavior of the components in temperature ranges in more detail.

The behavior of the components is stable only between the desired ranges. This is a very important aspect because although the measurement can be done at the ends supported by the sensor, the error will be much higher and enormously increases the deviation to small variations in ambient temperature. Therefore, always choose or build a circuit with components having the best performance as shown in the optimal region of the graph of the datasheet. Also, always respect and observe that the input voltages do not exceed the specified limit, otherwise it will be irreversibly damaged.

Note

To know more about ADC visit http://en.wikipedia.org/wiki/Analog-to-digital_converter.

Once we have the temperature sensor, we have to *extract* the data you need. Therefore, we make a simple calculation supported in the datasheet. As we can see, we feed 5V to the sensor, and it return values between 0 and 1023. So, we use a simple formula that allows us to convert the values obtained in voltage (analog value):

Voltage = (Value from the sensor) x (5/1024)

Note

The value 1024 is correct, as we have said that the data can range from 0 (including zero as a value) to 1023. This data can be strange to some, but in the world of computer and electronics, *zero* is seen as a very important value. Therefore, be careful when making calculations.

Once we obtain a value in volts, we proceed to convert this measurement in a data in degrees. For this, by using the formula from the datasheet, we can perform the conversion quickly. We use a variable that can store data with decimal point (double or float), otherwise we will be truncating the result and losing valuable information (this may seem strange, but this bug very common).

The formula for conversion is *Cº = (Voltage -0.5) x 100.0*.

We now have all the data we need and there is a better implementation of this sensor, in which we can eliminate noise and disturbance from the data. You can dive deeper into these issues if desired. With the preceding explanation, it will not be difficult to achieve.

Near Field Communication

Near Field Communication (**NFC**) technology is based on the RFID technology. Basically, the principle consists of two devices fused onto one board, one acting as an antenna to transmit signals and the other that acts as a reader/reciever. It exchanges information using electromagnetic fields. They deal with small pieces of information. It is enough for us to make this extraordinary property almost *magical*.

Note

For more information on RFID and NFC, visit the following links:

- http://en.wikipedia.org/wiki/Near_field_communication
- http://en.wikipedia.org/wiki/Radio-frequency_identification

Today, we find this technology in identification and access cards (a pioneering example is the Document ID used in countries like Spain), tickets for public transport, credit cards, and even mobile phones (with ability to make payment via NFC).

For this project, we will use a popular module **PN532 Adafruit RFID/NFC** board. Feel free to use any module that suits your needs. I selected this for its value, price, and above all, for its *chip*. The popular PN532 is largely supported by the community, and there are vast number of publications, tools, and libraries that allow us to integrate it in many projects.

For instance, you can use it in conjunction with an Arduino a Raspberry Pi, or directly with your computer (via a USB to serial adapter). Also, the PN532 board comes with a free **Mifare 1k** card, which we can use for our project.

You can also buy tag cards or even key rings that house a small circuit inside, on which information is stored, even encrypted information. One of the benefits, besides the low price of a set of labels, is that we can block access to certain areas of information or even all the information. This allows us in maintaining our valuable data safely or avoid the cloned/duplicate of this security tag.

There are a number of standards that allow us to classify the different types of existing cards, one of which is the Mifare 1K card (You can use the example presented here and make small changes to adapt it to other NFC cards).

Note

For more information on MIFARE cards, visit http://en.wikipedia.org/wiki/MIFARE.

As the name suggests, this model can store up to 1KB (Kilobyte) information, and is of the EEPROM type (can be rewritten). Furthermore, it possesses a *serial number* unique to each card (this is very interesting because it is a perfect *fingerprint* that will allow us to distinguish between two cards with the same information).

Internally, it is divided into 16 sectors (0-15), and is subdivided into 4 blocks (0-3) with at least 16 bytes of information. For each block, we can set a password that prevents reading your content if it does not pose. (At this point, we are not going to manipulate the key because the user can accidentally, encrypt a card, leaving it unusable). By default, all cards usually come with a default password (`ffffffffffff`).

The reader should consult the link http://www.adafruit.com/product/789, to continue with the examples, or if you select another antenna, search the internet for the same features. The link also has tutorials to make the board compatible with other devices (Raspberry Pi), the datasheet, and the library can be downloaded from Github to use this module.

We will have a look at this last point more closely later; just download the libraries for now and follow my steps. It is one of the best advantages of open hardware, that the designs can be changed and improved by anyone.

Note

Remember, when handling electrical components, we have to be careful and avoid electrostatic sources.

As you have already seen, the module is inserted directly on the Arduino. If soldered pins do not engage directly above then you have to use Dupont male-female connectors to be accessible other unused ports. Once we have the mounted module, we will use the library that is used to control this device (top link) and install it. Refer to Chapter 5, *Arduino and Sensors*, of this book that provides you with a step-by-step procedure. It is very important that you rename the library to download, eliminate possible blanks or the like. However, the import process may throw an error.

Once we have this ready and have our Mifare 1k card close, let's look at a small example that will help us to better understand all this technology and get the **UID** (**Unique Identifier**) of our tag:

```
// ###########################
// Author : Jorge Reyes Castro
// Arduino Home Security Book
// ###########################

#include <Wire.h>
#include <Adafruit_NFCShield_I2C.h>
// This is the Adafruit Library

//MACROS
#define IRQ  (2)
#define RESET (3)

Adafruit_NFCShield_I2C nfc(IRQ, RESET); //Prepare NFC MODULE

//SETUP
void setup(void){
  Serial.begin(115200);  //Open Serial Port
  Serial.print("###### Serial Port Ready ######\n");
  nfc.begin(); //Start NFC
  nfc.SAMConfig();
  if(!Serial){      //If the serial port don´ work wait
    delay(500);
  }

}
//LOOP
void loop(void) {
  uint8_t success;
  uint8_t uid[] = { 0, 0, 0, 0};  //Buffer to save the read value
  //uint8_t ok[] = { X, X, X, X }; //Put your serial Number. * 1
  uint8_t uidLength;

  success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid,
&uidLength); //   READ
  // If TAG PRESENT
  if (success) {

    Serial.println("Found card\n");
    Serial.print(" UID Value: \t");
    nfc.PrintHex(uid, uidLength);  //Convert Bytes to Hex.
    int n = 0;
    Serial.print("Your Serial Number: \t");
    // This function show the real value
    // 4 position runs extracting data
    while (n < 4){
      // Copy and remenber to use in the next example * 1
      Serial.print(uid[n]);
      Serial.print("\t");
      n++;
    }
    Serial.println("");
  }
  delay(1500);                //wait 1'5 sec
}
```

Now open a serial port from the IDE or just execute the previous Python script. We can see the output from the Arduino. Enter the following command to run the Python code:

```
$ python SerialPython.py
```

The output will look like this:

```
###### Serial Port Ready ######
Found card

UID Value:      0xED 0x05 0xED  0x9A
Your Serial Number:    237    5    237    154
```

So we have our own identification number, but what are those letters that appear above our number? Well, it is hexadecimal, another widely used way to represent numbers and information in the computer. It represents a small set of characters and more numbers in decimal notation than we usually use. (Remember our card with little space. We use hexadecimal to be more useful to save memory). An example is the number 15, we need two characters in tenth, while in decimal just one character is needed `f` (`0xf`, this is how the hexadecimal code, preceded by `0x`, this is a thing to remember as this would be of great help later on).

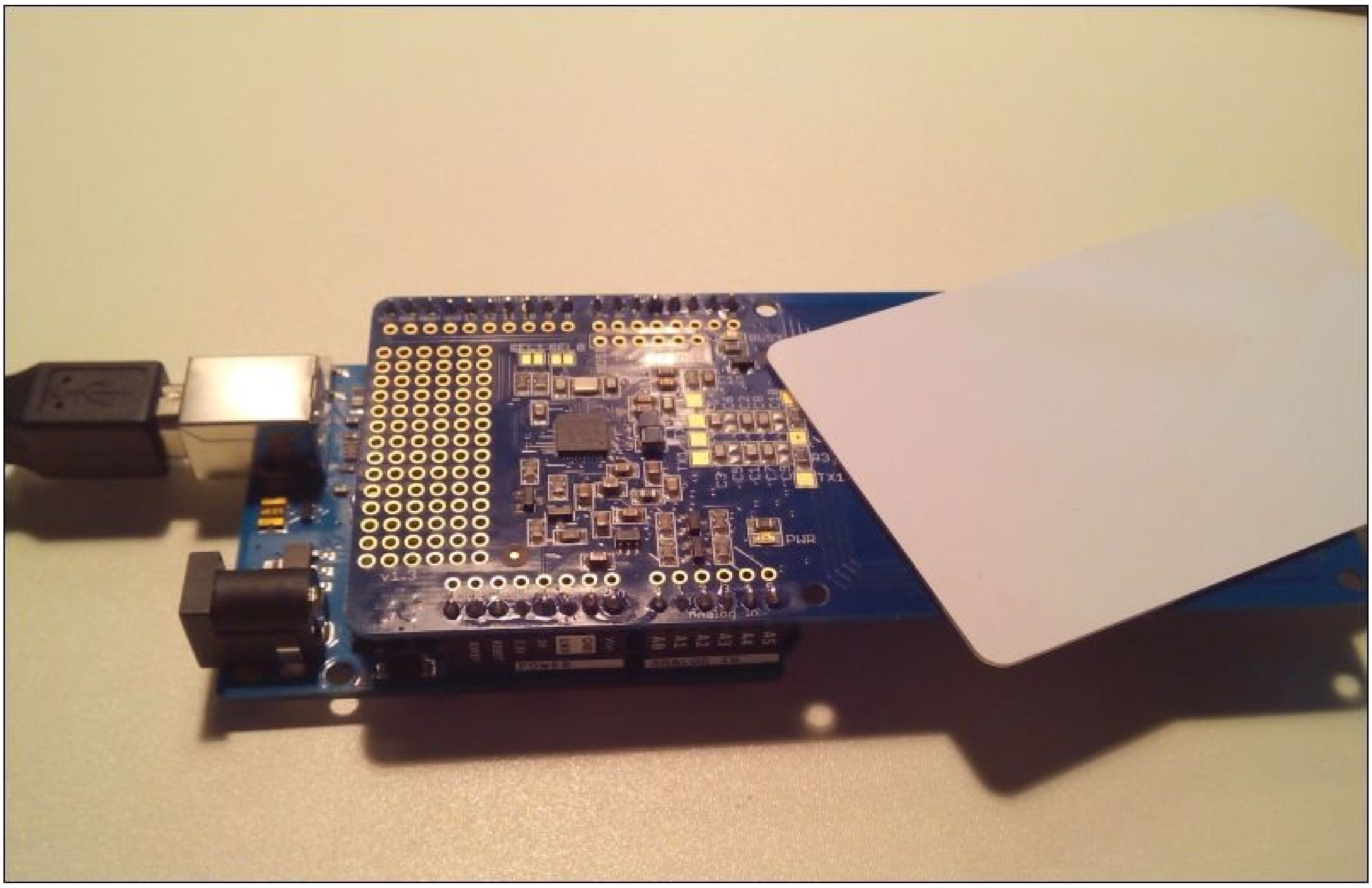
Open a console and the Python screen. Run the following code to obtain hexadecimal numbers, and will see transformation to decimal (you can replace them with the values previously put in the code):

Note

For more information about numbering systems, see http://en.wikipedia.org/wiki/Hexadecimal.

```
$ python
>>> 0xED
237
>>> 0x05
5
>>> 0x9A
154
```

You can see that they are same numbers.



RFID/NFC module and tag

Once we are already familiar with the use of this technology, we proceed to increase the difficultly and create our little access control. Be sure to record the serial number of your access card (in decimal).

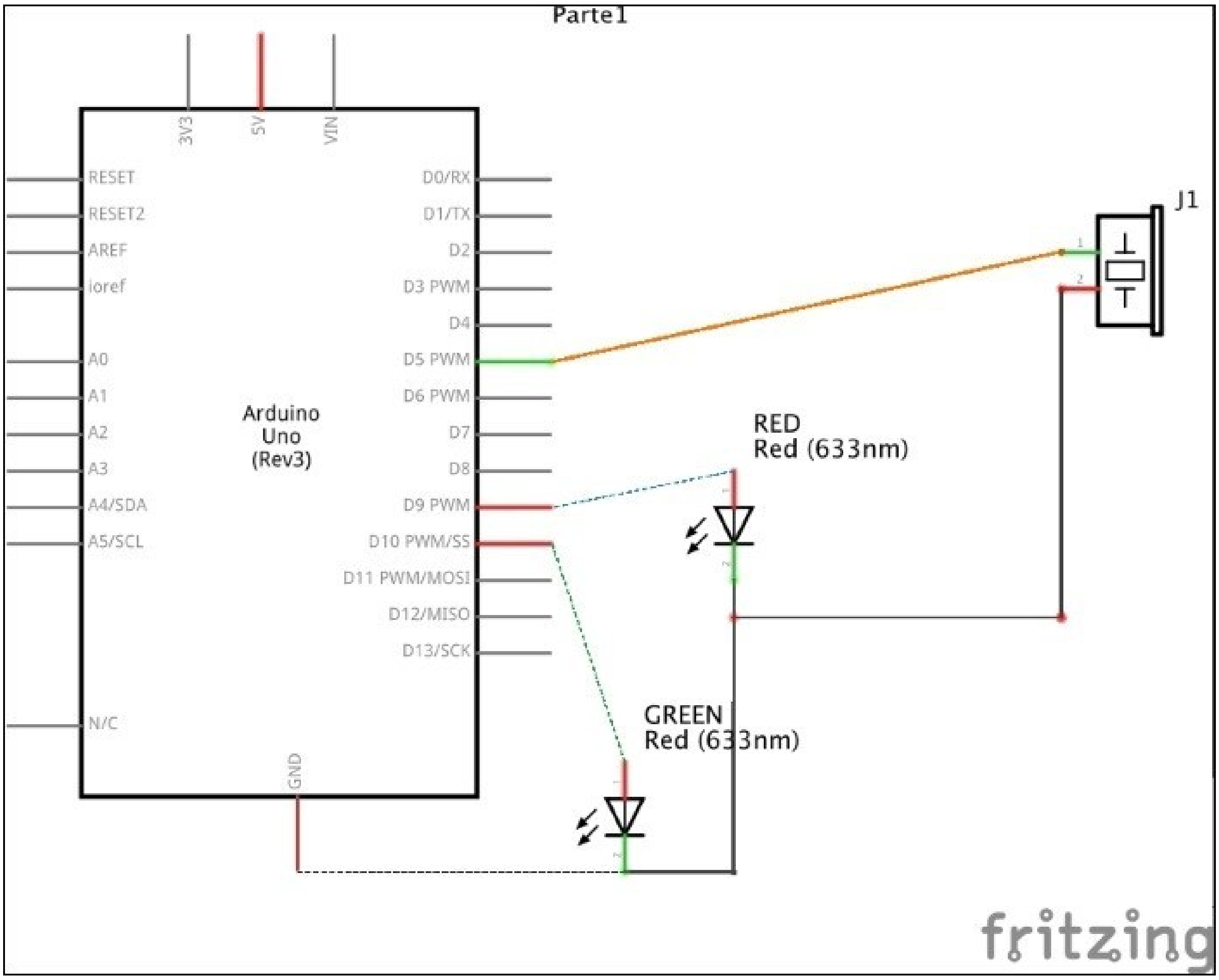
We will make this little montage focus on the use of NFC cards, no authentication key. As mentioned earlier, there is a variant of this exercise and I suggest that the reader complete this on their part, thus increasing the robustness of our assembly. In addition, we add a LED and buzzer to help us improve the usability.

Access control

The objective is simple: we just have to let people in who have a specific card with a specific serial number. If you want to use encrypted keys, you can change the ID of an encrypted message inside the card with a key known only to the creator of the card. When a successful identification occurs, a green flash lead us to report that someone has had authorized access to the system. In addition, the user is notified of the access through a pleasant sound, for example, will invite you to push the door to open it.

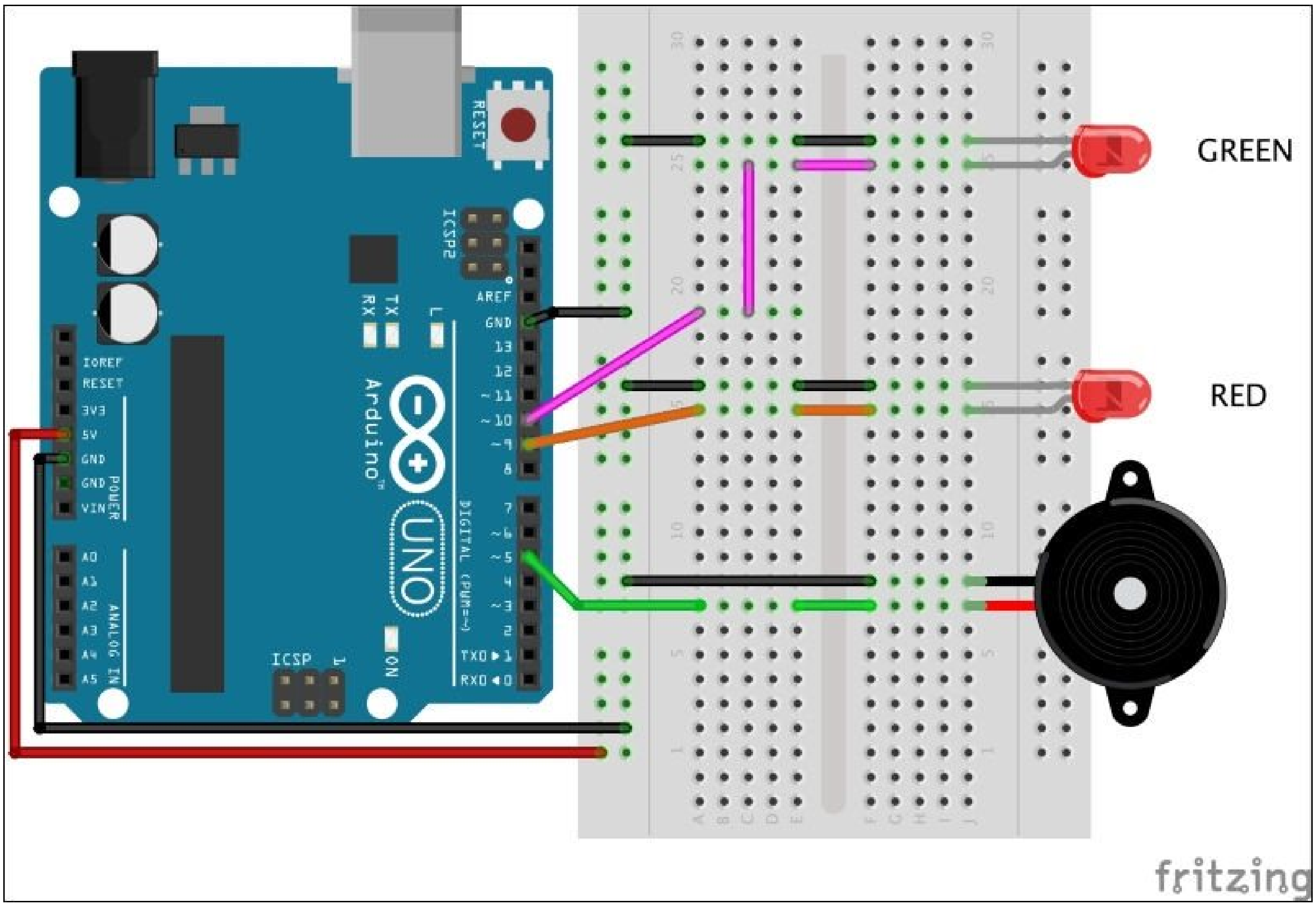
Otherwise, an alarm is sounded repeatedly, alerting us of the offense and activating an alert command center (a red flash that is repeated and consistent, which captures the attention of the watchman.)

Feel free to add more elements. The diagram shall be as follows:



Our scheme – Image obtained using Fritzing

The following is a much clearer representation of the preceding wiring diagram as it avoids the part of the antenna for easy reading:



Our design - Image obtained using Fritzing

Once we clear the way to take our design, you can begin to connect all elements and then create our code for the project. The following is the code for the NFC access:

```
//#############################
//Author : Jorge Reyes Castro
//Arduino Home Security Book
//#############################

#include <Wire.h>
#include <Adafruit_NFCShield_I2C.h> // this is the Adafruit Library

//MACROS
#define IRQ  (2)
#define RESET (3)

Adafruit_NFCShield_I2C nfc(IRQ, RESET); //Prepare NFC MODULE

void setup(void) {
  pinMode(5, OUTPUT); // PIEZO
  pinMode(9, OUTPUT); // LED RED
  pinMode(10, OUTPUT); // LED GREEN
  okHAL();
  Serial.begin(115200); //Open Serial Port
  Serial.print("###### Serial Port Ready ######\n");
  nfc.begin();     //Start NFC
  nfc.SAMConfig();
  if(!Serial){     // If the Serial Port don't work wait
    delay(500);
  }
}

void loop(void) {
  uint8_t success;
  uint8_t uid[] = { 0, 0, 0, 0};
  //Buffer to storage the ID from the read tag
  uint8_t ok[] = { 237, 5, 237, 154};
  //Put your serial Number.
  uint8_t uidLength;

  success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid,
&uidLength); //READ

  if (success) {
    okHAL();                // Sound "OK"
    Serial.println("Found card\n");
    Serial.print(" UID Value: \t");
    nfc.PrintHex(uid, uidLength);
    // The Value from the UID in HEX
    int n = 0;
    Serial.print("Your Serial Number: \t");
    // The Value from the UID in DEC
    while (n < 4){
      Serial.print(uid[n]);
      Serial.print("\t");
      n++;
    }

    Serial.println("");
    Serial.println(" Wait..\n");      // Verification
    int m = 0, l = 0;

    while (m < 5){
      if(uid[m] == ok[l]){
      // Compare the elements one by one from the obtained UID card that
was stored previously by us.
    }

    else if(m == 4){
      Serial.println("###### Authorized ######\n"); // ALL OK
      authOk();
      okHAL();
    }

    else{
      Serial.println("###### Unauthorized ######\n");
      // NOT EQUALS ALARM !!!
      authError();
      errorHAL();
      m = 6;
    }
    m++;
    l++;
  }
}

delay(1500);

}

//create a function that allows us to quickly create sounds
// alarm ( "time to wait" in ms, "tone/power")
void alarm(unsigned char wait, unsigned char power ){
 analogWrite(5, power);
 delay(wait);
 analogWrite(5, 0);

}
// HAL OK
void okHAL(){
  alarm(200, 250);
  alarm(100, 50);
  alarm(300, 250);
}

// HAL ERROR
void errorHAL(){
  int n = 0 ;
  while(n< 3){
  alarm(200, 50);
  alarm(100, 250);
  alarm(300, 50);
  n++;
}
}

// These functions activated led when we called.
// (Look at the code in the upper part where they are used)
// RED - FAST
void authError(){
  int n = 0 ;
  while(n< 20){
    digitalWrite(9, HIGH);
    delay(500);
    digitalWrite(9, LOW);
    delay(500);
    n++;
  }
}
// GREEN - SLOW
void authOk(){
  int n = 0 ;
  while(n< 5){
    digitalWrite(10, HIGH);
    delay(2000);
    digitalWrite(10, LOW);
    delay(500);
    n++;
  }
}

// This code can be reduced to increase efficiency and speed of execution,
// but has been created with didactic purposes and therefore increased
readability
```

Once you have the copy, compile it, and throw it on your board. Ensure that our creation now has a voice, whenever we start communicating or when someone tries to authenticate. In addition, we have added two simple LEDs that give us much information about our design.

Finally, we have our whole system, perhaps it will be time to improve our Python script and try the serial port settings, creating a red window or alarm sound on the computer that is receiving the data. It is the turn of reader to assimilate all of the concepts seen in this day and modify them to delve deeper and deeper into the wonderful world of programming.

Summary

This has been an intense chapter, full of new elements. In this chapter, we learned to handle technical documentation fluently, covered the different type of signals and their main differences, found the perfect component for our needs, and finally applied it to a real project, without forgetting the NFC. This has just been introduced, laying the foundation for the reader to be able to modify and study it deeper it in the future.

In the next chapter, we will come up with a design, the requirements, and identify the installation environment and the measures needed to develop it.

Chapter 4. Designing Your Own System

The aim of this chapter is to teach you certain concepts through a practical approach, rather than digging deep into the theory. It is an attempt to show you a different mindset or a way of working rather than a step-by-step approach.

In this chapter, we'll cover the following:

- Designing a system from scratch and looking through all its phases
- Identifying the needs
- Creating a first draft or a prototype
- Installing, using, and maintaining the system
- Electrical laws and principles

Designing a project

In the previous chapter, we created a small prototype, a system access control, but this was simple to assemble, where all the components were together, and on a breadboard, a rare thing in the real world. Also, we had to have some *control* over the system. It lacked autonomy and depended on the connection to the computer, and so on.

Therefore, we will begin to find out how to create a good design installation, where we will tackle all the possible problems that can be found throughout the creation, deployment, and subsequently, the exploitation.

Let’s start explaining these last three words—*creation*, *deployment*, and *exploitation*—which are the most important points when creating a solution.

Getting ideas for design

At this point, we must be able to identify the real problem we face. By this, we mean being able to perform an exercise in abstraction and discern between the apparent problem and the real problem. What I mentioned earlier may seem obvious, but only when a good identification exercise is done successfully is it possible to make the following points.

For example, imagine we want to know the number of people who pass through the door at any specific point in time. With such a simple *idea* in mind, we can come with different solutions, but few meet the proportionality (for the need/solution, this idea should be understood as an illustrative abstraction, since we are not able to quantify these terms). Returning to the previous point, one solution may be to have a simple laser and detector but what happens if multiple people go through the door all at once? The beam is then cut only once and those ten people will be counted as just one person passing, another possible option is to add a camera and do a visual count using facial recognition software, which is obviously expensive and complex, and if the location of the installation is not properly lighted, the image processing may not work properly.

We must gather all possible information from our environment, peculiarities, and desires of usage, all complete to the expectation of the solution and only then do we move on to the next phases.

To know the spatial distribution and distances in the environment where we want to work is a mandatory aspect in designing a system. Devote enough time exclusively to this aspect in a practical example.

Note

This guide can be applied to almost any industry where we have to tackle a problem. It is therefore advisable that these practices for such scenarios be followed.

Creating a design

Having identified the needs and features of the environment, we will start with the design phase.

Begin with the first draft of the design you have in mind. Do not give up if you have to redesign your deployment on more than one occasion, be aware of electric lines and points, and of the water piping around your location of installation.

At this point, you may have to choose between a number of factors, such as if to include the use of cable or not, and choose between the two - a wireless design or a wired one (which were explained in the first chapter). Also, we will be brushing up on their characteristics and disadvantages, hence we won't go into detail at this point, but remember that there is a possibility of interference and therefore some elements might not work properly.

Finally, we introduce a new concept, known as potential drop, which is the effect that occurs when electricity runs through a wired body or a body causing a decrease in voltage. This effect increases with the length, thickness, and properties of the conductor section, thereby affecting the values carried in them. Remember that the sensors operate within a range of values and behavior may vary or go crazy if it gets out of the range of values.

Deploying the project

Once you have carried out testing on a small circuit in our laboratory, proceed to deployment. If we decide to perform an installation, we must look for the materials necessary to perform the installation, such as the number of sensors, tools, and various equipment (tape, wire, and so on).

It also should include the number of people needed for this, the time, and the possible complications that may arise. Imagine we need access to a roof or area that is difficult to access, and we need to do it safely, you need to carry all important tools and equipment to prevent yourself from mishaps such as falling off the ladder (also you should never work alone in such cases).

It is advisable to carry out *debugging* from our facility. What this means is that we have to test the correct operation before moving on to the operational phase, including certain aspects that will never materialize or even hardly occur. This is the philosophy of reassessing constantly and expecting the unexpected.

Using the installation

Once you have completed the preceding steps, proceed to use and enjoy a running system. The operational phase will be the longest in the life of our creation. So where several things can happen, maybe we need to extend it because it is not enough, repair any item or give it a basic maintenance. In the example here, we understand this point better.

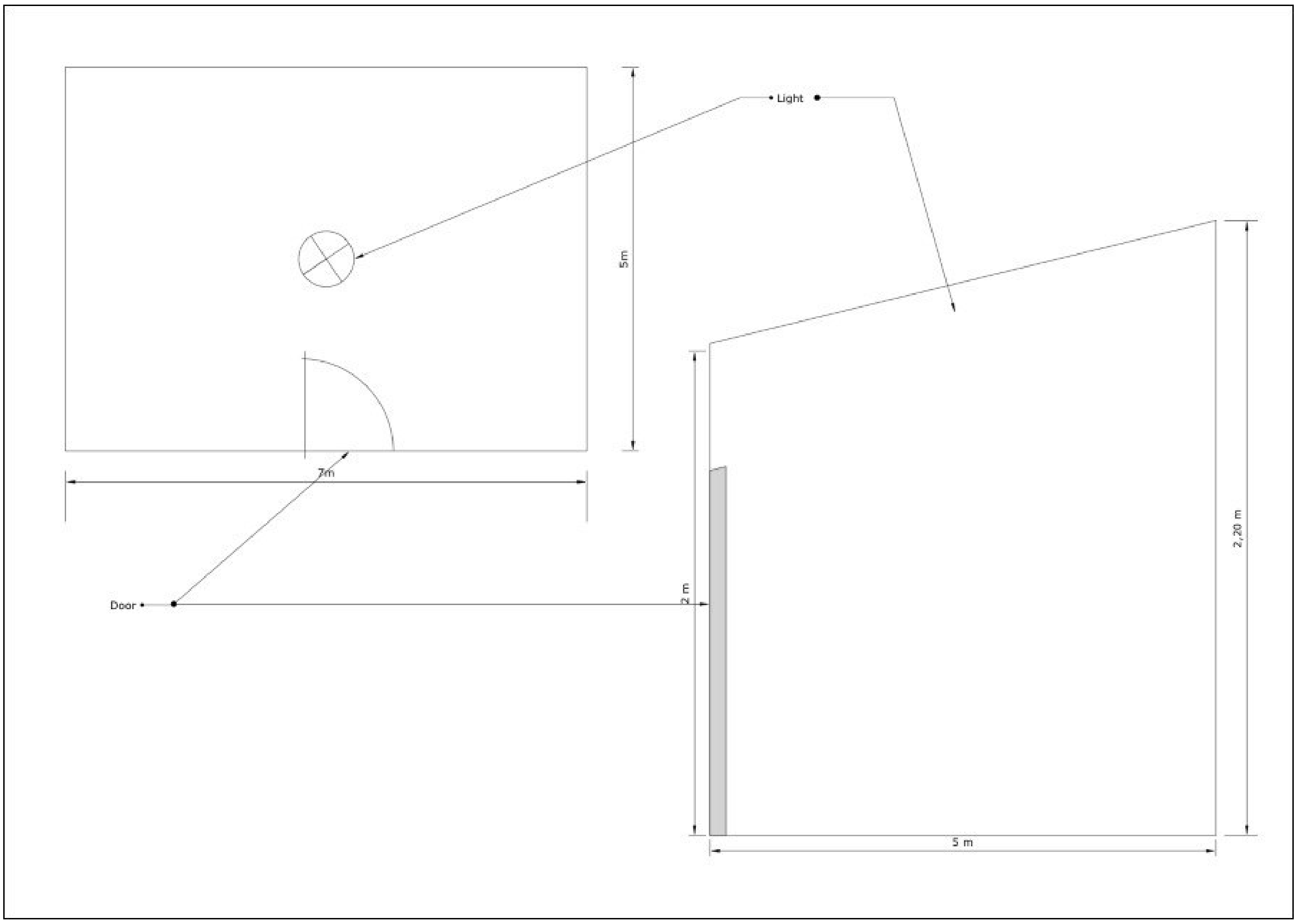
An example

After quickly going through the theoretical part, let's try to make an example that covers the basic features for you to practice and internalize how to approach these problems.

Let's say we want to create a system that allows us to turn off and turn on a light, and we remotely know their status lets us use an LED, which you know are low consumption devices and are protected against damages. This same example can be created using a relay and a light bulb rated at 220V. I have already revealed the first point – a need or problem to solve.

Step	**Comment**
Request	The status of a light and managed remotely
Design	Outlines and drafts

We will create a small *map area* and collect all possible information about it.



An example of a room

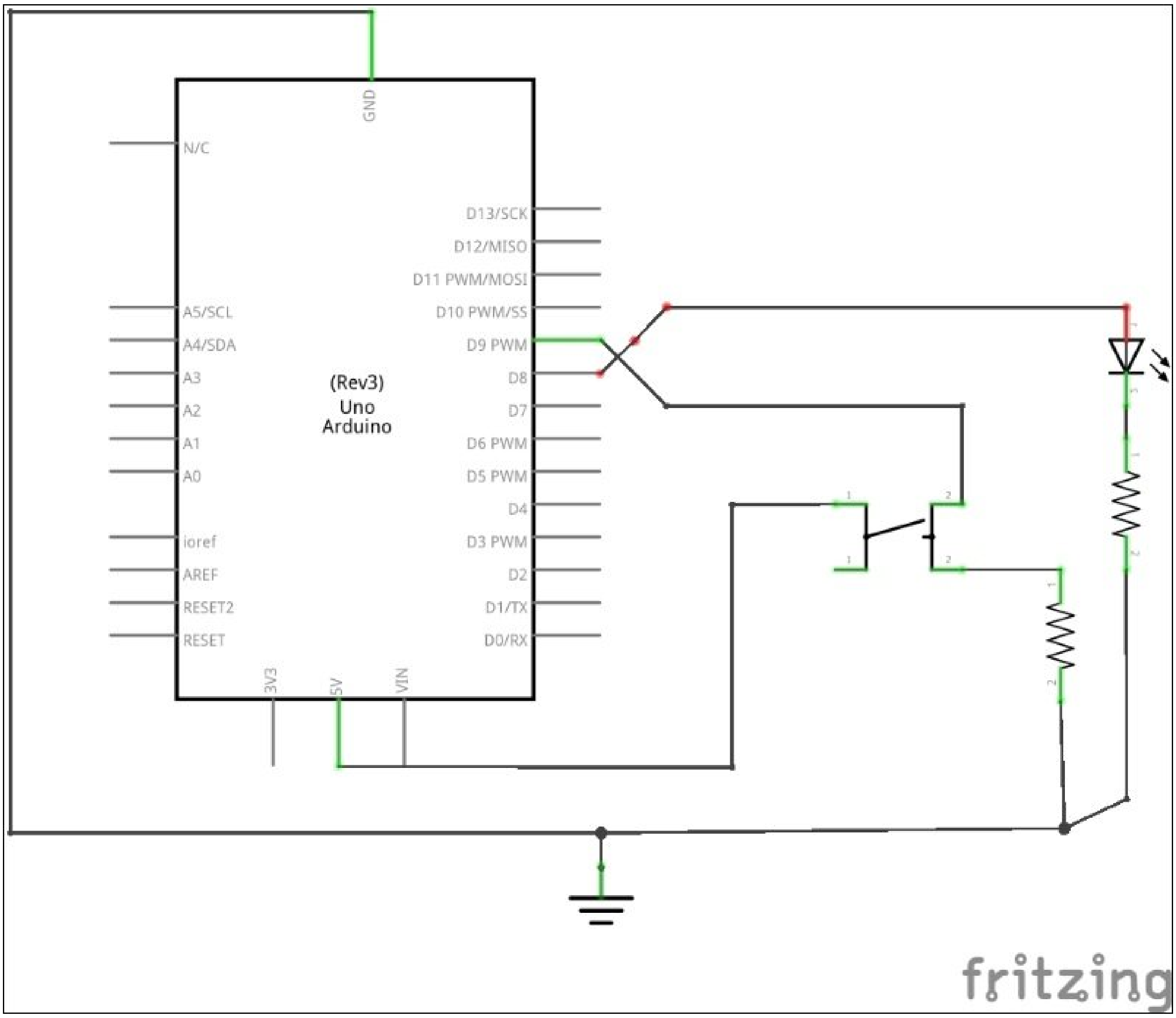
Note

SketchUp: A very useful and free software to make drawings and 3D models (including 3D printing) tool is this. It is highly recommended that you are familiar with this software (or any other of your choice) before you start to design your project. Go to the following link to download SketchUp: http://www.sketchup.com/download

In the preceding diagram, you can see we have a little room in which we want to control the overhead light, this same example can be done with components that work on higher voltages and power such as a household lamp or a light sensor (to know our lights are illuminated at night, assuming we want to simulate presence in a room or simply an automatic irrigation or any other sensor we add).

We have also identified a small outlet electric current and a WiFi network so that we can install our micro controller and a WiFi module or add a Raspberry Pi and an external battery. Use this example when we introduce a series of tools to then create something else that meets your needs.

The next step is to create a small electrical schematic for this project:



Our design - image obtained using Fritzing.

What we see in the preceding schematic is a microcontroller connected to a 220 ohm resistor, which is then connected to an LED, a button, another resistance (10 kΩ) associated with the button , and finally a computer/Raspberry Pi which will execute the script in Python.

The Python script *reads* the status of the button and when it changes the script will send an e-mail. The system tells us in real time what is happening.

To continue the example here, `smtplib` needs to be installed in the Python library, it can be installed with the `pip` command followed by the library name; for more information, visit the following links:

- https://pypi.python.org/pypi/pip
- https://docs.python.org/2/library/smtplib.html

The following is the Arduino code:

```
//CODE
//#############################
//Author : Jorge Reyes Castro
//Arduino Home Security Book
//#############################
// 9 <- Input
// 8 -> Output
//#############################

//Global Variable
int on = 0;

//Setup
void setup(){
// Initial Serial port communication

Serial.begin(9600);


//PINMODE

pinMode(8, OUTPUT);
pinMode(9, INPUT);
qa
//DEFAULT

digitalWrite(8, LOW);
digitalWrite(9, LOW);

//LOW = 0 logic
//HIGH = 1 logic
}

//LOOP

void loop(){
// Store de value of the buttom
intsensorVal = digitalRead(9);
    // not preset
    if (sensorVal == 0){
    }
// preset --> change status

    else{ // if sensorVal = 1 (we want to change the status)

     if(on == 0){
digitalWrite(8, HIGH);
//Turn ON
     on = 1;
// Value for Python
Serial.println("Y");
     }

     else{ // if sensorVal = 1 & on = 1
digitalWrite(8, LOW);
     // Turn OFF
Serial.println("N");
     // Value for Python
     on = 0;
     }

    }

//Wait 500
  delay(500);
//return to loop

}
```

In the previous code, we have associated the state of a switch with a variable and then sent its status through the serial port.

Let’s pursue this with the Python code (the same code can be used on any platform that supports this language, including Raspberry Pi), so we can remotely manage our Arduino:

```
# CODE
###########################
# Author : Jorge Reyes Castro
# Arduino Home Security Book
###########################
# Library
import time  # to use slepp
import smtplib # send email
import serial # SerialPort

#Prepare Serial Port
sport = serial.Serial('/dev/tty.usbmodemfa1311', 9600)

# SERVER CONFIG
###############
# I use gmail, but you must put your own data
# search on internet for your account data
###############
# SMTP =>OUTSITE or SENDMAIL
PORT = '587' #Put yout server port
SERVER = 'smtp.gmail.com'
TO = '...@gmail.com' # addressee
USER = '...@gmail.com' # from
PASS = '***********' # password
SUB = 'HAL'          # subject
TEXT = 'THE LIGHT IS ON !!!' # Text to send
###############
#Function that help you to send mail
###############
def notify():
    print "try to send"
    server = smtplib.SMTP(SERVER,PORT)
    # Try to connect
    server.ehlo()
    server.starttls()
    server.ehlo()
    # Try to authenticate
    server.login(USER, PASS)
    # Create a message
    notifycation = 'To:' + TO + '\n' + \
    'From:' + USER + '\n' + \
    'Subject:' + SUB + '\n' + \
    '\n' + TEXT + '\n\n'
    # Show you in the serial port the message
    print notification
    server.sendmail(USER,TO,notification)
    # Mail Send
    # Close the connection
    server.close()
    print "close"

###############
# MAIN
###############
while True:
    readed = sport.readline()
    # Allways read
    if readed[0] == 'Y' :
    # If Arduino send Y send email
        notify() # Upper function
    else:
        print "Waiting"
    time.sleep(0.5)
    # Wait 1/2 seg

#########
```

Note

WARNING!

E-mail accounts have certain security measures to not accept requests from unusual origins, for example, blocking or regional or even continental requests.

It may be that the process of *login* to run this script fails, and a warning is printed to the console. If so, simply access your account via a web browser and disable the measure (do not use your personal account, create an account for testing or development, this will preserve its security and allow you to enjoy this funny script).

Now, just execute the previous Python script with the following command, and you can see the output on the Arduino UNO:

```
$ python SerialEmail.py
```

The output will look like this:

```
________________________
try to send
To: …@gmail.com
From:…@gmail.com
Subject:HAL
```

```
//THE LIGHT IS ON !!!

Close
```

Finally, after making a montage on the breadboard and testing everything, we can move on to the next step, mounting. The following are the steps to be observed:

Step	**Comment**
• Request	The status of a light and managed remotely
• Design	Outlines and drafts
• Installation	To put all together
• Exploitation	Use and maintenance

For this part, we require some basic tools, in addition to insulating adhesive tape, a box to protect our assembly, if we do not want to connect it to the mains or batteries. Arduino is powered by a USB input that you can power. You may want this type of solution because of its low cost, easy handling, and durability.

If you elected an assembly with real bulbs or high voltage elements, disconnect the power supply and use elements of insulating protection.

Proceed to make a quick installation to begin the testing phase *in situ*. But, *will my model work once installed? Do I have to wait for someone to press buttons to know the status of light? What if it does not work? What if it is failing?* Well, all these questions are frequently thought of in such projects, therefore, it is time to introduce a new concept, which has been previously mentioned succinctly, debugging and logging.

Debug is the action by which we are able to *cleanse* (or inspect) our code or implementation. To put it in a more simple way, what you do is actually run the code in a rugged manner and study their behavior at all times (the variable values should change while debugging).

Why should I use debugging

Well, in the preceding example, imagine that you have had compilation errors, or after copying the code exactly, the system behaves in the wrong way or you are not able to understand what is happening in your system.

Well, all we have are the logs, which are small informational messages that can be used to monitor the status code and certain parameters.

The main difference between the *logs* and *print* is that the former should only be shown to certain people, who maintain services, and on occasion, in maintenance and debugging. Imagine that the data displayed on the console is accessible to users and thus appears as a message indicating that the process *X* is running, or that the variable *Y* is initialized with a value of `1`. This will damage the aspect of our standard solution and confuse the user.

For this, we use the following structure. I urge you to modify and improve the preceding code and add debug options based on the basic outline that I will comment:

```
Import logging
# Library

logging.info('INFO')
# Show normal information in a standard format
logging.warning('WARNNING')
# Show normal alert in a standard format
logging.error('ERROR')
# Show error alert in a standard format in red color
logging.debug('DEBUGGIN')
# Show information to debbuger
# Only if you call the script with the "-d" option.
```

With these options, we can connect quickly to our programs and consult certain values or *draw* a variable that interests us and is comfortable and elegant. This finishes the design and installation part.

The basic principles of electricity

If you are very young or are new to making electronic projects, you may not be familiar with certain principle terms of electricity. So let’s dedicate this last part of the chapter to learning them, plus also find out about the basic materials that are necessary to test your projects. To explain certain aspects needed for the development of the book, it is advisable to consult the links provided.

Voltage

The exact term for the voltage is potential difference, or the value obtained subtracting the voltage between a pole and a battery or the end of any component of the circuit.

All elements of a circuit have some amount of *resistance* to current flow, including generators that generate electricity, because for various physical principles when we create or transform one form of energy into another, we are doing work and therefore *consume* layoff of that energy generated or transported. The unit of measurement is volts.

Note

To read more on voltage, visit: http://en.wikipedia.org/wiki/Voltage

Conductor resistance

As mentioned in the previous section, resistance is the opposition to the passage of electric current through a body, this factor depends on the intrinsic properties of the material (metal is a good conductor, while plastic is not), and cross-section component and length, among others. The unit of measurement is the ohm. (A resistor has colored bands on its body that it allow to recognize the value of its internal resistance in ohms).

Note

To find out more about resistance, visit:
http://en.wikipedia.org/wiki/Electrical_resistance_and_conductance

Current

Lastly, as a result of applying the mathematical formula of the previous two terms, we obtain the intensity of electricity. That is, the amount of current that is going through a body with a known resistance. The unit of measurement is **Ampere**.

Note

To find out more about current, visit: http://en.wikipedia.org/wiki/Electric_current

Ohm’s law

All are related by Ohm’s law, which allows us to calculate one of the three elements knowing the other two components, the following is the mathematical expression:

$V = I * R$ (Voltage = Intensity * Resistance)

Note

To find out more about Ohm’s law visit http://en.wikipedia.org/wiki/Ohm%27s_law.

Joule’s law

An irreversible effect where a conductor is heated depending on the load to pass through it and its properties. This aspect is sometimes simple and must be internalized and taken into account whenever working with electronic elements, where they have gathered a large number of electronic components in a confined space, for example, processors or chips.

To find out more about Joule’s law visit http://en.wikipedia.org/wiki/Joule_heating.

Resistors and capacitors

Once you have finally understood the aforementioned terms, we will continue with resistors, these are electronic components created to *define* or hinder the flow of electricity through a circuit, an example is the resistance of the previous assembly, where the LED cannot be under much voltage and is kept safe with a 220 ohm resistor so the LED is not burnt out.

Besides these, curious elements are capacitors, which have the property of storing unbelievable loads for a short time.

Note

Surely, the reader has heard of capacitors, and even about very specific types of them, they consist of tantalum, which is a mineral or rather dried solution of two minerals, also known as coltan.

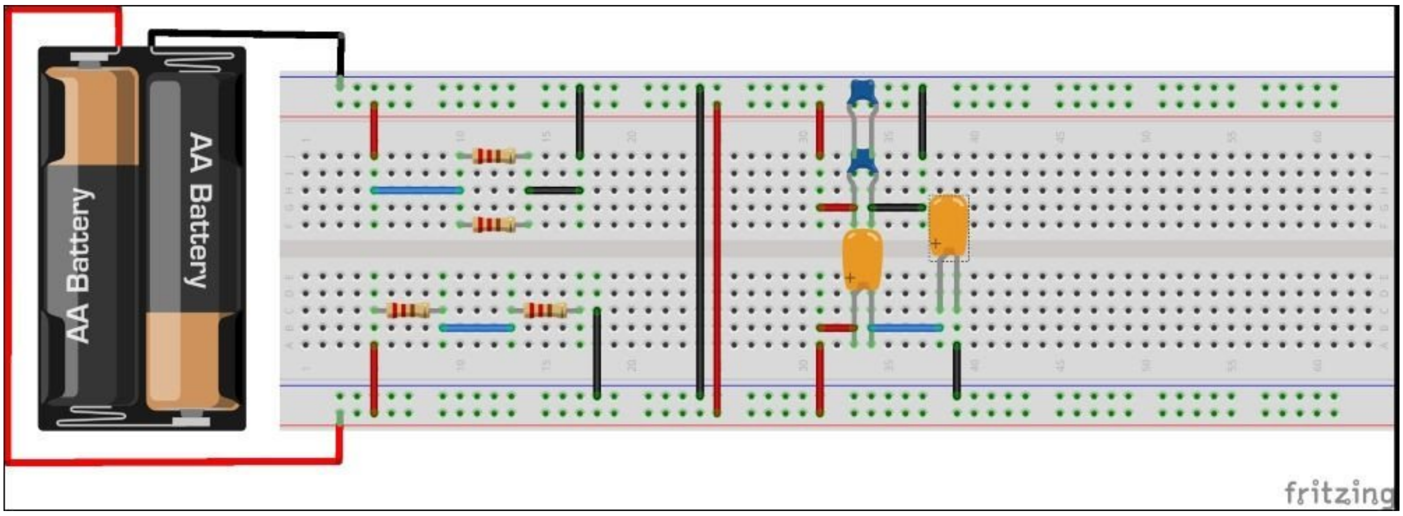
I recommend the reader to read more about this element that has amazing properties in the world of electronics, and is unfortunately scarce and often embroiled in conflict. While some traders have followed policies to ensure that its products are created with *conflict minerals*.

Visit http://en.wikipedia.org/wiki/Tantalum to find out more about tantalum, and visit http://en.wikipedia.org/wiki/Coltan to know more about coltan.

With capacitors we can create a circuit that can *delay* the signal (although circuits' complexity is beyond the scope of this book, the interested reader can get more information by clicking on the following link:

http://en.wikipedia.org/wiki/RC_circuit

Once we have the necessary knowledge, we will conduct a series of measurements on a test circuit, shown as follows:



A test circuit - image obtained using Fritzing.

As we can see, we have four circuits, which start from the top left-hand side and end in the lower-right corner, reading circuits is similar to reading in English. We have a *parallel* circuit by the arrangement of resistors together. They are crossed by the same voltage, but have a different intensity. Then we have a parallel circuit of ceramic capacitors with resistors in series, *in-line* with a different voltage drop but with an identical intensity. The circuit ends with tantalum capacitors in series.

We can, if we want to know the value of these components, use two different methods together or separately:

Theoretical analysis

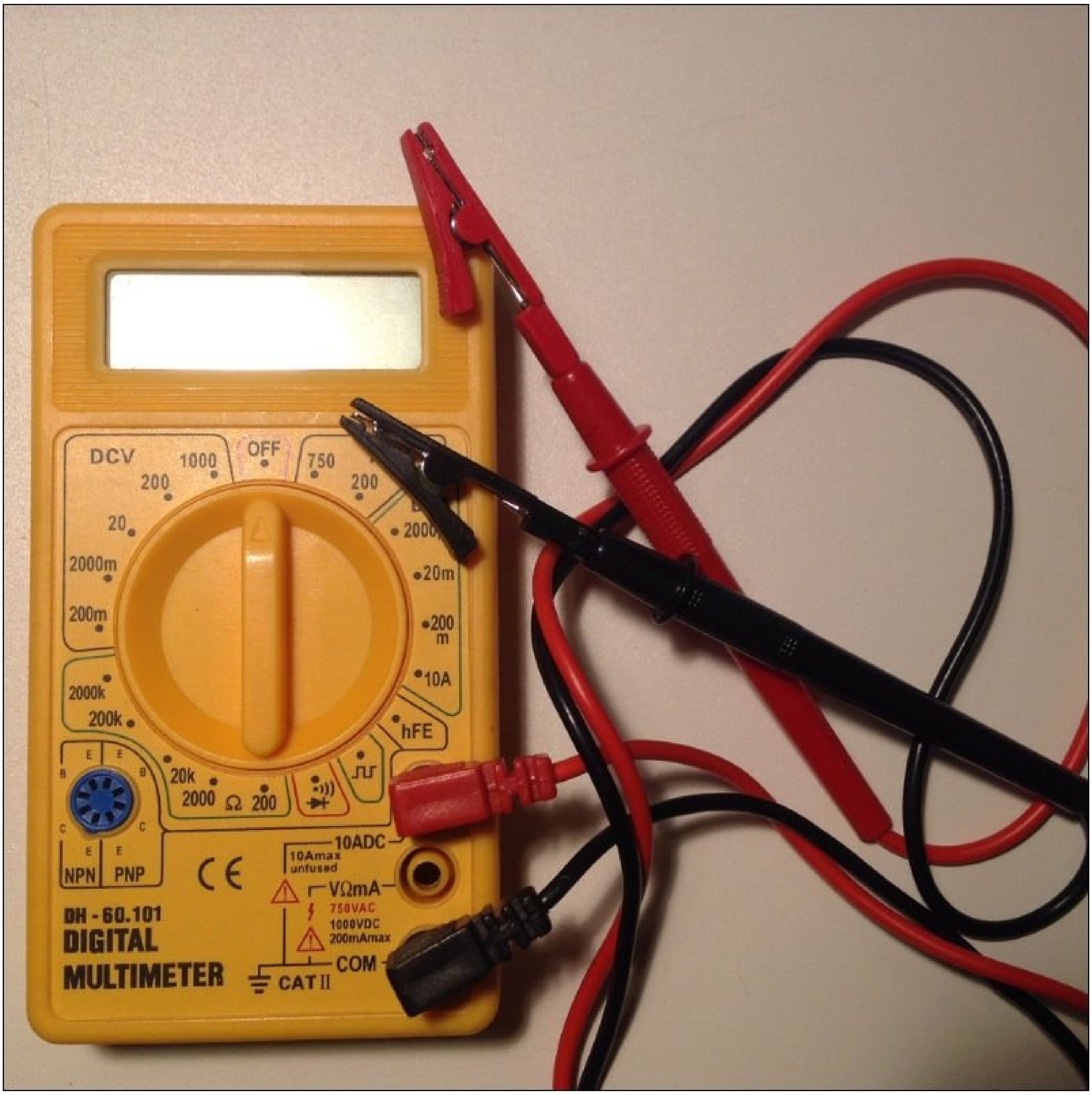
Theoretical analysis allows us to calculate the *approximate* values of our circuit (this value will be different from the actual value, by numerous factors, such as manufacturing processes, product characteristics, the dry stone imperfections that the manufacturer admits, and ambient temperature). For this, we take the help of the already known Ohm's law and also of the following mathematical formulas:

- **Series circuit**: When in this arrangement, the resistors should join and get an *equivalent* resistance in that part of the circuit we remove a section and replace it with a single resistor with a value equal to that resulting from the following calculation:
  - For resistors: $R = R1 + R2 + R3$
  - For capacitors: $1/C = 1/C1 + 1/C2 +1/C3$

  If you're dealing with capacitors, we have to make the sum of 1 divided by the value of the capacitance (in nanofarads or microfarads, which is the unit of measurement of these stored charges).
- **Parallel circuit**: In a parallel arrangement of components, the circuit is just the opposite of what series connection is. We will combine the strengths and resistances calculated with the *reverse* of its value (1 divided by the value of the resistance). The following is the formula for resistors and capacitors in series connection:
  - For resistors: 1/R = 1/R1 + 1/R2 + 1/R3
  - For capacitors: C = C1 + C2 + C3

The digital multimeter

Perhaps, at this point, the reader is thinking that the preceding method is tedious or boring. However, it is the basis for good design and knows the limitations of the technology and its expected behavior. Relax! We have a powerful tool called multimeter, which is capable of calculating large number of values such as voltage, current, and resistance, besides electrical continuity among many other values. The following image shows a digital multimeter:



A digital multimeter

As you can see, the multimeter has an LCD screen that shows the measurement readout, a wheel switch to select the kind of measurement you need, and has two probe leads to connect the meter to the physical component to measure its value. It can be purchased at electronic stores and even hardware stores. It doesn’t cost much and is very handy in certain situations and even essential.

Note

To find out more about multimeters, visit https://en.wikipedia.org/wiki/Multimeter.

Once you have understood these concepts, it is recommended to return to the test circuit we saw previously and try to calculate equivalent resistance and capacitance, first numerically and then using multimeter.

Note

Finally, I do not want to miss the opportunity to recommend a small pocket tool, which is a smartphone application called **ElectroDroid** pocket to all passionate electronics enthusiasts and hobbyists. Always use this when you need to make some calculative checks, or when you are in any doubt regarding basic electronic principles and calculation, you do not need to memorize formulas any more. To find out more about this app, visit www.electrodroid.it.

Summary

We have gained enough knowledge so far to create our own project design from scratch, identify its needs and requirements, concepts, and then transfer the design on paper. Later we can create a prototype and finally install it in the real world. We have given our Arduino the ability to send real-time tasks being performed and the state of a sensor.

In the next chapter, among other things, we will work with the libraries on how to create, import, and modify to increase the power of our code. Furthermore, we will also learn to integrate more sensors and circuit elements such as MOSFETs, and learn to control them.

Chapter 5. Arduino and Sensors

If you have gotten this far, I congratulate you for your performance and eagerness to learn. In this chapter, we will learn interesting topics such as:

- How to create, use, and find libraries
- New sensors and components for our system, such as LCD
- More hardware
- Debugging

Besides all this, we will strengthen certain aspects. From a more practical perspective, we will use concepts that we've previously seen.

As you recall, resistors and capacitors are the essential elements in the circuits. Now, I want to show you a *new* property. It is an input to the RC (resistor-capacitor) circuits.

We will discuss this point in more detail. I encourage you to find out more about these issues and visit the linked information. Capacitors can store an electrical charge, measured in multiples of farad, usually picofarads or microfarads, and when subjected to an electric current, they can store a charge (energy). A very interesting property that once fully charged, impedes the flow of current, thus acting as a temporary switch. The energy source that begins to discharge is eliminated in the circuit. This feature is used to remove electrical oscillations (the ups and brownouts), thereby producing a more stable signal.

Resistors, can be have fixed and variable values. The fixed resistor has a unique ohmic value which is color-coded and printed on its sides, while the variable resistors can change their resistance with an adjustable gear. Resistors are open to the passage of current, limiting the flow of electrical supply. This helps a particular sector of the circuit to protect the elements, as sometimes higher voltages will destroy the components.

If we combine a resistor and a capacitor in a circuit, we can create **filters**, which prevent the passage of certain frequencies or levels. We need not delve more into this concept now, but in the future when we design our circuit, it can be very helpful. You have to respect the existence of these elements, together or separately, and even understand why we use these or why we use resistors that have different characteristics within the same kind of circuit, depending on their placement. For example, we will be able to soften or eliminate interference signals or noise (widely used in digital electronics, radio systems, and signal processing).

Note

You can refer to the following links for more information on filters:

- http://en.wikipedia.org/wiki/High-pass_filter
- http://en.wikipedia.org/wiki/Filter_(signal_processing)
- http://en.wikipedia.org/wiki/Bode_plot
- http://en.wikipedia.org/wiki/RC_circuit

After reviewing this concept, we will move quickly on to see what a library is, how we can create it or find it on the Internet, and why it is useful.

The code library

In programming, we always look for code optimization, reuse, and simplicity. Under these three principles, we create all our projects and get a *hold* of the code, so it is easy to integrate a piece of code in to another project, reducing effort and time.

Suppose that we want to print a message via the serial port and we want to do it in a concrete way in a special format.(We did this in one of our first examples, where we created a function that will spend a number of arguments and the function that will print it). Instead of repeating the same code many times, we create a *module* that will help us print a message in a format desired as many times as you like. And once that module is created, it will be as easy as writing a line of code to call it.

Using code libraries prevents us from again creating the same function, but with a different outcome. But what if we have a group of functions, which we want to reuse in many projects, or simply want to separate the code base to maintain it or improve it? For this we have libraries, which are a set of functions grouped in a file that share certain properties.

We know what a library is, now we are realizing its potential use and versatility. Another use of this is to facilitate the integration of external elements to our code; for example, we want to add an LCD to our project, but do not want to write hundreds of lines of code for it separately, while the others have already been created (as you can see, libraries open a new world where we can share pieces of code from other people with the same interests as us).

Making your own library

To understand what a library is, we will create our own library. For that, we need to follow these steps (independent of the OS we are working from):

1. Create a folder without spaces in its name, and with a different name to any existing folder on your computer (or you might end up in replacing critical system folders). The name should not be same as the other library. For example, `Jlib`. (If you prefer Unix-based systems, you can do all of this from the terminal itself, with the commands as shown:

   ```
   George:~$mkdir ./Jlib// New folder
   ```

   ### Note

   You can use a text editor such as Notepad Plus Plus (Notepad ++) and Sublime Text, these are code-friendly text editors:

   http://notepad-plus-plus.org/

   http://www.sublimetext.com/

2. Do the following steps:

   ```
   George:~$cd ./Jlib          //Enter the folder
   George:~$touch ./jlib.cpp ./jlib.h ./keywords.txt
   George:~$nano ./jlib.cpp
   ```

The following is the code for the `Jlib.cpp` file:

```
//CODE
//#############################
//Author : Jorge Reyes Castro
//Arduino Home Security Book
//#############################

#include "Arduino.h" // To use the functions of Arduino
#include "Jlib.h"  // Own library

// Jlib Class::function
void Jlib::format(char* text){    #f1
  Serial.println(" --###################-- ");
  Serial.print("   ");
  Serial.print(text);
  Serial.print("\n");
  Serial.println(" --###################-- \n");
}

void Jlib::start(){                #f2
  format("Author"); // Call upper function
  format("J.R.Castro");
  format("Starting Program");
}
```

In the `Jlib.cpp` file, we defined two functions belonging to the `JLIB` class. The first function (`f1` as we can see in the comment of the code) receives a pointer to a text, while the second (`f2`) receives no parameter and is based on the foregoing, to show a particular message.

Now, make another file called `Jlib.h`:

```
George:~$nano ./jlib.h
```

Enter the following code into the `Jlib.h` file by using the text editor:

```
//CODE
//#############################
//Author : Jorge Reyes Castro
//Arduino Home Security Book
//#############################

#include "Arduino.h"     // To use the functions of Arduino
#ifndef Jlib_h           // If not define Jlib
#define Jlib_h           // We prevent import more than once

class Jlib{              // Define the class
  public:               // Function accessible for all
  void start();
  void format(char* text);
};

#endif
```

`Jlib.h` is a head file that serves to define headers and to specify the preceding functions belonging to a particular class; remember that we are programming in a language with object-orientation (OOP). To find out more about object-oriented programming, visit http://en.wikipedia.org/wiki/Object-oriented_programming.

Once you are done creating the header file, create a file named `Keywords.txt`, enter the following text in to the `keywords.txt` file:

```
Jlib      KEYWORD1
format    KEYWORD2
start     KEYWORD2
```

This file is optional, but we have decided to include a library in the generic way a library is made. You must include the name of the library (`KEYWORD1`) and the function within it (`KEYWORD2`). In this way we are stating that the library is composed of a series of functions (allowing us to call them from the IDE and recognize them as functions). Always be aware of the case sensitive (do not add comments or extra characters).

Once we have all these files created, we will proceed to add it them our IDE. To do this, we must perform the following steps:

1. Open the Arduino IDE.
2. Navigate to the upper toolbox to **Sketch** | **Import Library** | **Add Library**.
3. Add the folder or ZIP file (which is identical to your folder, but compressed for easy sharing).

4. Look at the Arduino console for possible errors.

   Note

   Now, we have a copy of the folder with the new library, for example, if we have defined the wrong library or want to make changes, we will delete that file and then import it back, so it is essential to check before for the integrity of our code. Visit http://arduino.cc/en/guide/libraries, to learn more about libraries.

5. Continue to create a simple program to use this new library, with the sole purpose of illustrating this process and knowing that the code does not perform any *spectacular* function. Use the following code:

```
//CODE
//#############################
//Author : Jorge Reyes Castro
//Arduino Home Security Book
//#############################

#include <jlib.h>        // include our library
Jlib obj;               // Create a Object

// SETUP
void setup(){
  Serial.begin(9600); // Open Serial Prot --> 9600 baud
  obj.start();               // Call the function of the object
}

//LOOP
void loop(){
// Call the function of the object
  obj.format("Hellow Library");
  delay(4000);     // Wait 4 seconds
}
```

As you can see, the code is simple. The important points to note are the importing of our library (`#include jlib.h`), an object (`obj`), and finally access to the functions of that object (`obj.start` and `obj.format`).

The output will look like this:

```
--###################--
     Author
--###################--

--###################--
   Jorge R.Castro
--###################--

--###################--
   Hellow Library
--###################--
```

With all that you know now, you can try to create, manage, and use the libraries, but what happens if you want a ready-to-use library? For this we have third-party code libraries.

Third-party libraries

In cases where we cannot create a library from scratch, or when we are only interested in running an element without the need to write many lines of code, test, and debug it. For these cases, we can go to websites of open source code, where people like you upload *free* code, for example **GitHub**.

Note

To find out more about GitHub, visit https://github.com/.

The next chapter will show you how the user can use GitHub and applications. For now, only mention that it is a repository of open-sourced code that anyone can view, read, and download to improve, fix, and share again.

To download a library from the webpage, just click on the **download zip** icon, then extract it. Once extracted, ensure that the folder does not include names that are not valid (spaces and non-alphanumeric characters). Once you have completed these steps you are ready to import the IDE, the same way as explained in the previous case.

Debugging the code

By *debugging*, we evaluate and analyze code at the time of its development with the only purpose of evaluating each of its component pieces of code.

A typical example is a variable that changes as you run your code. If we want to know its state at a particular time, we need a tool that allows us to monitor it. For this, there is debugging.

You will be surprised at how many times we think that a piece of code is behaving in a particular way and need to debug it, but their behavior is just the opposite (usually it is the case when loops or conditionals are used).

There are several programs that are able to perform these tasks, but few or none are free to use or are of a proprietary type, so I opted for the following option. It is easy to use and you can keep using the same program:

```
//CODE
//############################
//Author : Jorge Reyes Castro
//Arduino Home Security Book
//############################
const boolean debug=1; // Invariable constant one see the compiled code
void setup(){
  Serial.begin(9600);
}
void loop(){
  if (debug == true)
  {
    Serial.print("YOU ARE DEBUGGING ME");
  }
  else
  {
    Serial.print(" NO DEBUG");
  }
}
```

As you can see, we have an elegant yet practical solution, where we can find the definition of a constant to which a value is assigned and if it is equal to true, we console the “logs” information that interests us. Otherwise, that part remains hidden. Although it’s present in memory (that occupies space), so perhaps a final version is optimal, remove these fragments to save space.

Note

As mentioned earlier, there are programs that can debug your code, but have not been selected for use here, because of their nature, feel free to investigate them and use programs such as Atmel Studio or Visual Studio.

More hardware

Let’s take a look at a practical example in a project, where we add more elements such as adjustable resistors, potentiometers (employed in appliances that you use daily, for example, the volume control on your radio), or LCDs which now everybody knows about and are present in almost every electronic device that humans use. Lets learn more about this hardware.

The LCD

We also know that the **Liquid Crystal Display** (**LCD**) is an illuminated surface (either by natural light or by an energy source), which has set of pixels in it. They usually consume low power and aren't pricy. The advantage of using them is that it is not necessary to create an expensive system with separate LEDs that are synchronized and compose characters. It integrates an internal circuit that allows us to communicate in a simple way to display characters.

Note that this is a convenient and economical way to convey a message to the user, and sometimes must perform a series of operations (a typical example is a soda vending machine).

Note

To find out more about LCDs, visit the following links:

- http://en.wikipedia.org/wiki/Liquid-crystal_display
- https://www.adafruit.com/datasheets/TC1602A-01T.pdf

The potentiometer

When we wish to vary the amount of current that passes in a circuit, we can use potentiometers. These are circuit elements that are resistors with adjustable parts, providing a fixed resistance range.

There are various types of potentiometers depending on their composition and behavior. The most basic behave linearly (but vary on a regular basis), while others are logarithmic (grow as a curve, which are widely used for audio equipment, an example is the DJ mixer).

Note

To find out more about potentiometers, visit http://en.wikipedia.org/wiki/Potentiometer.

Semiconductors

Before learning about the **MOSFET** (**Metal Oxide Semi-conductor Field Effect Transistor**), we first must know what a **transistor** is. We should learn about its technical aspects briefly.

A semiconductor is an electronic component that comprises of a set of elements that are of the conducting and non-conducting nature, in fair proportion, to obtain special properties. They are active elements in a circuit which is able to switch to different conditions and are designed for a specific outcome. In short, they act as switches in a circuit. Transistors are found in every digital application, such as your computer, it contains a billion transistors on its CPU chip alone.

Within the family of semiconductors, there is the simplest of all devices – the **diode**. Suppose that electricity is like the flow of water through a pipe, but you just want to get a small amount of it, for this we use a valve (which allows us to halt the flow). A diode basically works like a water tap, it lets water flow in one direction and not the other way back.

The diode behaves the same way and allows us to fix an amount of electricity flowing in a circuit. In addition, valves (and diodes) prevent current from backing down the drain, this is very useful when your circuit is handling AC power.

On the other hand, transistors (and including MOSFET), elements with four contact leads (although often the user will only see three) act as *gates*. (Within the scope of this book, lets not get into much detail). Let me introduce a similar example. Suppose we have a dam, and the water symbolizes the electric current, we can close or open the dam by adjusting the flow rate of water entering. But to accomplish this task, we have to supply electricity to the door to get it to move. Something similar happens with transistors, if the voltage support receives a small charge and at a particular time, it is capable of increasing *flow* or *amplifying* the force used.

Note

To find out more about semiconductors, visit the following links:

- http://en.wikipedia.org/wiki/Transistor
- http://en.wikipedia.org/wiki/MOSFET
- http://en.wikipedia.org/wiki/Semiconductor_device

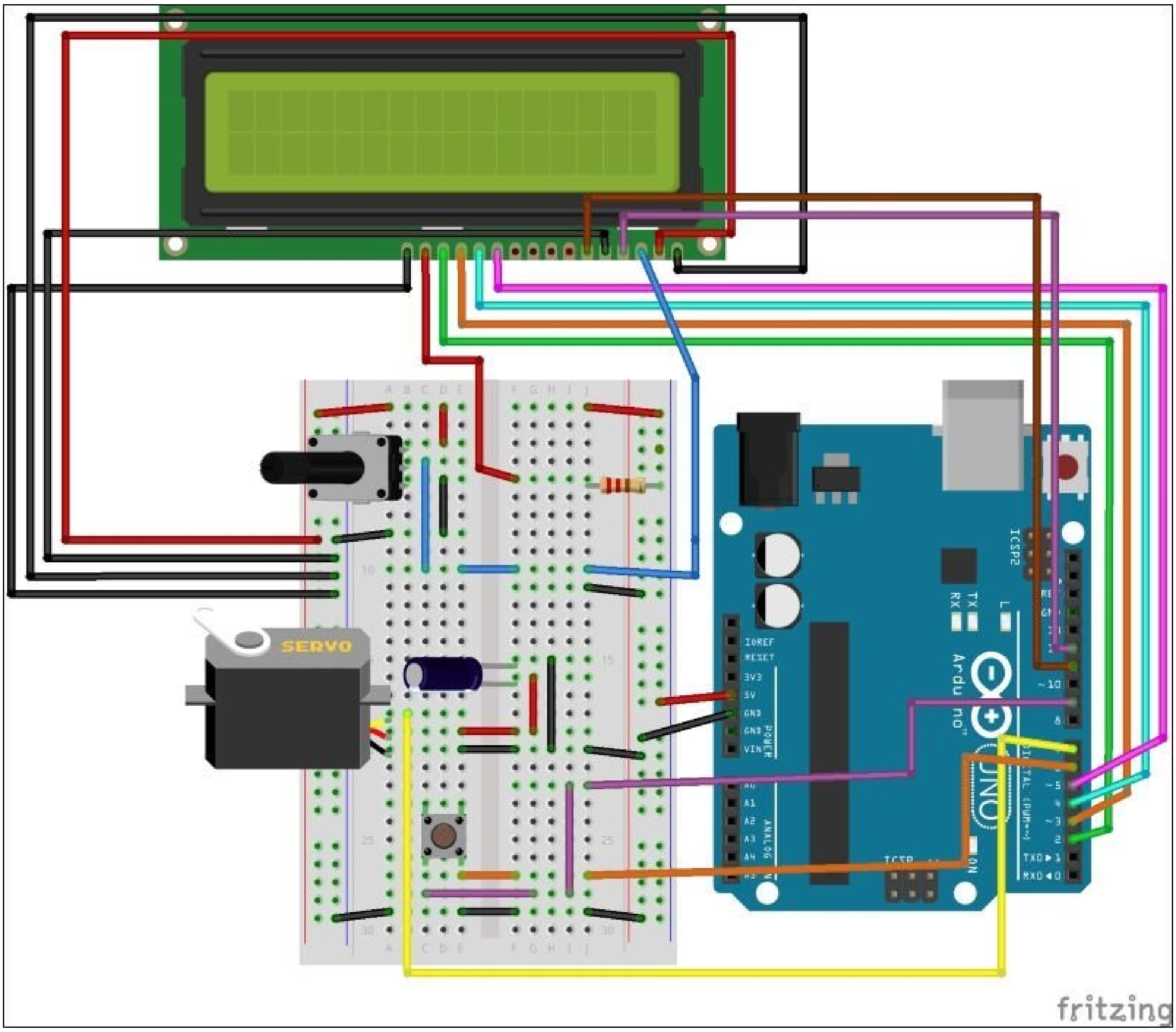
Its application is very wide, and represents a revolution in the technology world. Under this same concept, logic gates were created, followed by processors of computers. In a single circuit (integrated), there are millions of these miniature elements.

A mini project

Before ending the chapter, let's build a mini project in which we can implement the many aspects seen so far. The idea is to create a system with stepper motors (this is a new element and consists of a small motor capable of turning its spindle to a particular prompted degree of rotation), allowing us to open or close a latch or around door accesses. A button will allow us to manage the door and an interface, with an LCD screen, to notify us about the status of the door (open/close).

We will use a potentiometer to adjust the LCD contrast, a capacitor to the servomotor (and thus soften the electrical signal that reaches it), and finally will have a port as an output and one input to read the status of a button, which will vary an internal value. Finally, we will include debug logs through the serial port.

The circuit schematic is as shown:

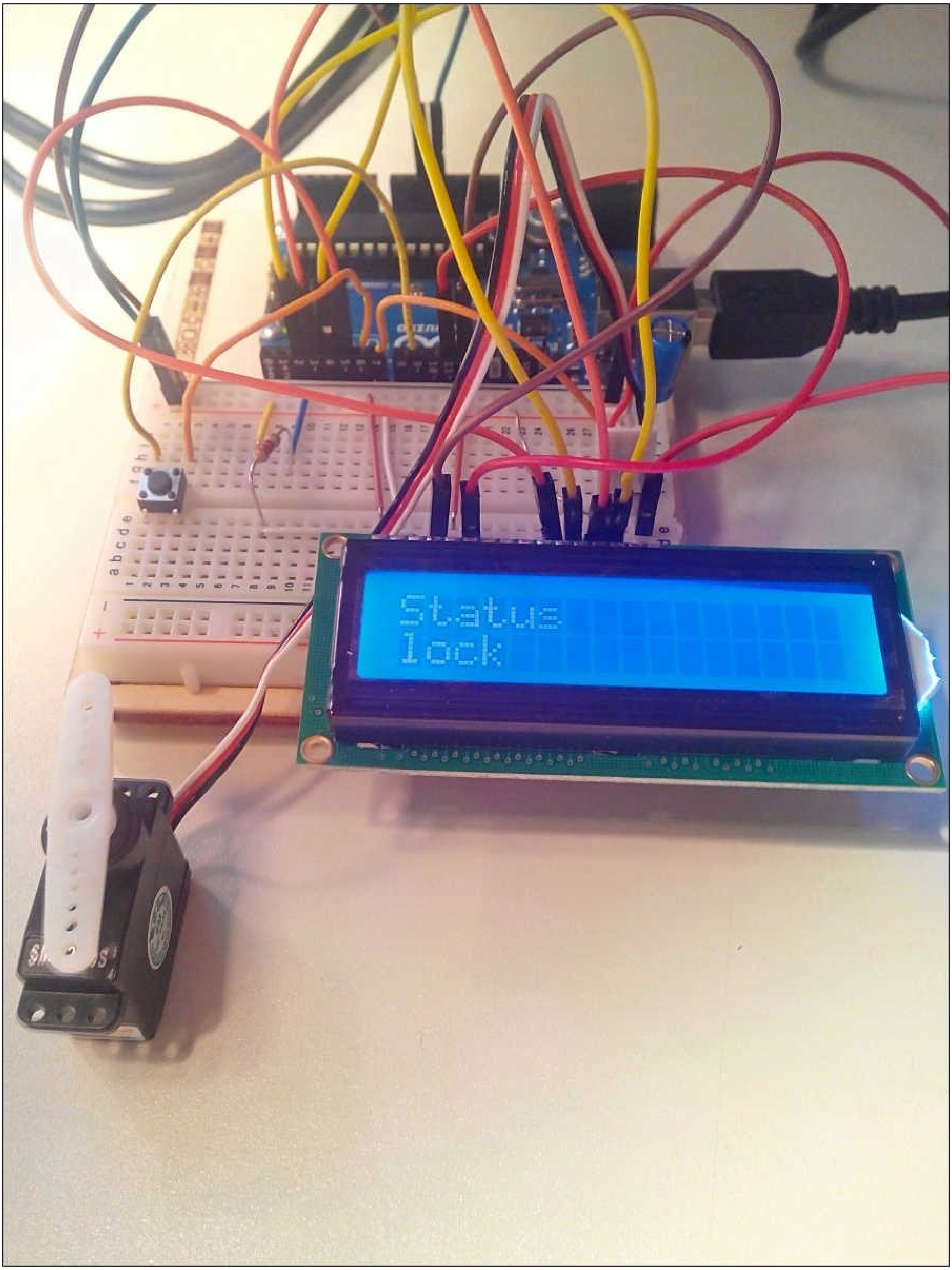


The LCD and servo motor mini project - image obtained using Fritzing

The code for the project is as follows, name the file as `LDCDoor.ino`:

```
//###########################
//Author : Jorge Reyes Castro
//Arduino Home Security Book
//###########################
#include <LiquidCrystal.h> // Libraryes
#include <Servo.h>

const int debug = 0; // If we want to debug put =1
LiquidCrystal lcd(12,11,5,4,3,2); //pin of Arduino
// Status latch
boolean lock = false;
// Button pressed
boolean button = false;
// type Servo
Servo latch;

// SETUP
void setup(){
  Serial.begin(9600);
  // button
  pinMode(6, INPUT);
  pinMode(9, OUTPUT);
  digitalWrite(9, HIGH);
  // Port 7
  latch.attach(7);
  // Lock by default
  latch.write(90);
// latch.write(degrees)
// 90 = Closed
// LCD
  lcd.begin(16,2); // 16 characters 2 rows
  lcd.print("Hi I'm your");
// move one line down to write
  lcd.setCursor(0,1);
  lcd.print("Security System");
  delay(1000);
}

void loop(){
  button = 0;          // As default
  lcd.clear();         // Clean Screen
  lcd.setCursor(0,0);  // beginning

  //show status latch
  if (lock = true){
    mprint("lock");
  }

  //show status latch
  if (lock = false){
    mprint("unlock");
  }
  // If pulse
  if (digitalRead(6) == HIGH){
    button = true;
  }
  if (button == true){
    latch.write(0); // put degrees = open door
  }
  if (button == false){
    if(lock == true){
      //closes after 2 seconds, if we leave it open
      latch.write(90);
      mprint("LOCKING");
    }
  }

  // Log
  if(debug == 1){
    Serial.print(button);
    Serial.print(lock);
  }
  delay(1000);
}
//My function print lcd
void mprint(char* text){
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(""Status"");
  lcd.setCursor(0,1);
  lcd.print(text);
}
```

This is how the project looks once made:



The LCD and servo mini project

Summary

With all this, we finish this intense chapter, in which we have seen how to create a library and import it before we further add hardware to our designs, learning to use each component. At this point, I encourage you to review all these concepts and put them into practice and also create your own library, and share it.

In the next chapter, you will learn about Git code and documentation, and how to share it. Additionally, we will also begin to create a GUI.

Chapter 6. Documentation and Version Control

By now, you must be able to design a project, build it, and even put it to work. However, there are several important points that have not been explored yet, and we will see them in this chapter. Some of the most important points are:

- Code documentation
- Version control
- Graphical User Interface (GUI)

These are among the most important points, without which a project can never be complete, and will be very difficult to maintain or update in the future.

Code style and documentation

Often, we forget to comment on the functionality of our code (either in the code itself or in a separate document) with regards to the uses of the features. There are numerous cases where major companies have had to spend many hours updating the code because of not having this in the documentation or comments. You may think that a good programmer can read the code fluently, as long as some basic style rules are followed and are specific to each language. An example is giving meaningful names to variables, and not repeating identifiers, or using very long and complicated names. However, not with respect to the indentations, (in Python) this cannot happen because our code will not work and the interpreter will give out an error message.

Since every comment is ignored by the compiler and interpreters, remember that only if you have the source code can you read reviews of the authors. This seems obvious, but many people forget that compilers look to optimize and eliminate blanks, comments, and other elements such as variable names. There are two main ways to document a project:

- **Documentation for the end-user**: This is a simple guide, which explains what each function does in the program, without going into very technical details. Remember that the end-user might not be familiar with programming and that the work of “our creation” is to facilitate a task.
- **Technical Specifications:** This is the document that really interests us more, and is more specific, clearly stating the purpose of the project, why it’s needed and how it works internally, and how it receives input parameters. It also documents the inputs and outputs of the code. It details the various modules composed and dependencies used (the libraries it uses, or additional software that is needed to run the code).

Documents can be prepared in many ways, as long as they are comfortable and easy to read. We can also use a web page (there are multiple ways to create a document with a favorite text editor and export it to HTML) or a PDF file. Most commonly, we can create a file named `README.txt`, in plain text that accompanies the source code.

When we transform a document to HTML, it should be stored in a chapter format and only be visible locally. This may surprise many readers, because in your web browser you can see the document without accessing the Internet. (You can test this by creating a simple HTML page later to open it locally.)

On the other hand, certain large-scale projects often require their own websites online, where the developers create a wiki of the documentation. Anyone can view such documentation, even if the solution is implemented. Anybody can write comments, raise questions, or even report a bug and suggest methods to improve.

You can now see that there are many ways to create documentation. The important part to remember is that you must document, comment on your code, and respect the rules of styling for the chosen programming language. Only then will your project be created correctly. Then you may share it with anyone, and by sharing it with the community, you will also allow others to make improvements and use your code. We’ll talk about this in the next section.

Note

There are more types of documentation. Visit the following links to learn more about two other popular ways of documenting code:

- http://en.wikipedia.org/wiki/Mercurial
- http://en.wikipedia.org/wiki/Apache_Subversion

Version control and Git

So far, we've only talked about creating projects that are locally stored only in *your* computer. However, what will happen if we make a change, after rewriting the source code, and then wish to return to a previous state? You can save a manual copy of each change to overcome this issue, but this is highly impractical.

In addition, many projects are developed in groups. Imagine that you and colleague are creating a program in your spare time. Every day one of you makes a change and sends it to the other by email to pick up from where the earlier left off. Every day you will have to download the file, replace the existing version, and so on. After doing this for quite a long time, you will realize that this routine is ineffective and tedious. To counter such issues, there is a comfortable and faster way to maintain a state of control over your code, and share it in a very effective way. For this, we are going to use **Git**.

Git is a small program installed on your computer that enables you to keep track of the versions of your code from a base, and also get different branches of development (we can develop two separate codes in parallel and then join them as one). To better understand this concept, think of the project as a tree. It started as a seed and it grows larger as the code increases. We can create different branches of the tree (different versions of the same project). If we want to restore the previous state, we can go back and choose another direction. We can also prune the tree (remove branches abandoned by developers).

Note

Git was created by Linus Torvalds, the creator of the Linux kernel (http://en.wikipedia.org/wiki/Linus_Torvalds). To learn more about Git, I recommend visiting http://en.wikipedia.org/wiki/Git_%28software%29, which explains in detail what Git is.

All of the revisions can be done locally, either within your computer, or on a local server (this option is widely used in universities and companies). It can even be done directly over the Internet, where everyone will be able to read and share the code.

Note

Warning!

There are several reasons why you shouldn't upload particular elements of an Internet-based project, such as the IP addresses of your servers (if you are working with applications to be connected to certain important servers).

Do not forget that malicious code can be found anywhere. Read the code before installing its instances. Even so, in some communities trust in *open* code is rare. If you detect malicious code, please report it to its source and have it removed from your system environment.

As mentioned before, this book will be about teaching practically with little complex theoretical content. To delve deeper into these contents and has a series of useful links that will help you in increasing your knowledge.

Let’s proceed to install Git and look at some simple commands in it, which are very useful, and gain some explanation of them.

Installing Git

The first step is to install the Git manager for different platforms, as follows:

- **Ubuntu/Debian**: from the shell command prompt, enter the following command line:

  ```
  GeorgeDebian:~$ sudoapt-git install git
  ```

- **Fedora**: from the shell command prompt, enter the following command line:

  ```
  GeorgeFedora:~$ sudoyuminstall git
  ```

- **Arch Linux**: from the shell command prompt, enter the following command line:

  ```
  GeorgeArch:~$ sudopacman -S git
  ```

- **OpenBSD**: from the shell command prompt, enter the following command line:

  ```
  GeorgeOpenBSD:~$ sudopkg_addgit
  ```

- **Windows and Mac systems**: download the GUI installer from the following websites for your respective OS:
  - Windows: http://git-scm.com/download/win
  - Mac OS: http://git-scm.com/download/mac

  This is necessary since the base in these systems does not have an installer through the console. (The user should be familiar with these installers and therefore they will not be explained.)

There are hundreds of items and different ways to use Git, either through a console or graphical user interface (GUI). If you want to delve deeper into the more advanced aspects, I recommend you research online. (In this part we only will explain its more general aspects.)

The steps are the same if you’re using GIT locally or via a server, so lets proceed to the explanations directly.

Note

Quick guide

In my opinion, the following is one of the most simple, comfortable, and well-explained guides on Git that can be found on the Internet. I recommend saving the link as it can be very handy:

- http://rogerdudler.github.io/git-guide/index.html

Creating a repository and sharing it with Git

We will have to create a folder, in which all the subfolders will be contained, or in other words, these folders will "hang" from a root folder (similar to the branches of a tree), and add it to Git (if you have not enabled the setting to view hidden folders, you will not see certain files that are mentioned, since Git hides certain elements such that it does not mess certain directories from the user's view. Enter the following commands in the terminal to create a Git repository:

```
George:~$ cd Newdirectory     //Change to the new directory

GeorgeArch:~$ gitinit         // Our new repository

GeorgeArch:~$ gitadd .        // Add to git all the subfolder (note the
ending point ".")

GeorgeArch:~$ gitcommint –a –m "Date: today -> First Commit"
// "is a free text"
```

With this, we have created our first Git repository. We specified that we want to mirror (`-m`) all changes made within that folder ("`.`"). Finally, we committed the changes, indicating a comment, date, and sustained changes.

At this point, we stored a copy on our machine, but the case maybe that we want to have a copy on a server; this requires following article in the next section.

Sharing a copy of your code

If we want to upload a copy to a server, we need to have access to that server with its corresponding account and password. I recommend that you research the different options for having a Git repository online, either private or public.

Note

To get more help with working on Git, visit the following sites:

- http://git-scm.com/videos
- http://git-scm.com/docs/git-help

Although the most recommended is the one that is free and public, where everyone can read your code, improve it, share it further, and, of course, respect the authorship of the code. The following are the methods for uploading and downloading the code:

- **Uploading the code**: type the following command line to upload the code online:

  ```
  George:~$ git push URL
  ```

- **Downloading the code**: type the following command line to download the code:

  ```
  George:~$ git pull URL
  ```

Git ignore

As we saw earlier, there are certain things that we do not want to be copied in the repositories for security or privacy, or to simply hide it for practicality. For this, there is an option in Git – ignore. I recommend visiting the two links given here which explain this point in more in detail and give examples for each language.

Note

To learn more about Git ignore, visit the following links:

- https://github.com/github/gitignore
- http://git-scm.com/docs/gitignore

Git clone

If you want to download a repository and then work on it, then this is very simple. We use the Git clone option. This command will download the code you want and you can create a local copy of the code uploaded by a person:

```
George:~$ git clone URL
```

With all this and the skills necessary to use version control, I recommend you study more on everything discussed up to this point. When a local repository is created and uploaded to an online space, it will have a small file named `test.txt`. Do not forget to create the basic documents of the `README.txt`. (This last step is assigned as homework – as I mentioned before, is good to document the code.)

Having gained a very clear understanding of these points, we will create a small project to help you get into the design of a small GUI.

The Graphical User Interface

In our everyday lives, we often use many applications, whether on our smartphones, tablets, or our computers at work or at home, and you will agree that almost none of us knows what is beneath the layers of GUI. Few people know the code needed to create those nice screens and interactive applications with buttons. When we want to know the weather in our city, we just look it up on our favorite weather application. We don't have to call functions or query anything any more, we just click on the icon that says *Madrid* or *London* and get the information we want.

All this has a **Graphical User Interface**, or a **GUI**, and our project will have a GUI too. We will start with something simple, and leave the bases if you want to refine.

In this case, we have chosen to create a simple web interface, based on the languages already introduced, Python and HTML to be the front-end of our code. Do not worry if you do not know HTML. Here, we will have a simple example. If you want to know more, we will provide material to create something better.

Note

In the following link, you may find numerous tutorials on different web languages. Here, you can access a series of tutorials that will guide you in using the code, and also access a small console where you can see the results of the code you try.

- http://www.w3schools.com/html/

For now, we will stick to an interface on the same computer, but we will include remote access to the interface in the next chapter. It is recommended for this case, and only during the demonstration verification, that the firewall is not blocking the port that we will be using (since we cannot access it from our web browser).

A mini project using the GUI

We are assuming that you already have **Python 2.7** installed and configured. Follow the steps in the order given here, without skipping any, to achieve your goal:

1. **pip**: this is an essential tool that helps when working with Python. This will help you download and install any book shop you need simply and comfortably. To do this, you must download the `get-pip.py` file, ensuring that it retains the file extension `.py`, and open a terminal and type:

   - On Linux: `sudo python get-pip.py`
   - On Windows: `get-pip.py`

   ```
   pip install -U setuptools
   ```

   To update the pip tools, type:

   Note

   Copy and save the code that appears when you open the following link in a document (use a text editor to paste the copied code and save the file) named `get-pip.py`:

   https://bootstrap.pypa.io/get-pip.py

   If you have any further questions about the installation, then you can consult this guide:

   https://pip.pypa.io/en/latest/installing.html

2. **Web.py**: once you're done with the previous step, you have to install the new library, called `web.py`, so use this command from the installed pip directory with administrative permissions:

   ```
   pip install web.py
   ```

Once you have everything ready, you can create a simple webpage that can be accessed from the local network. Take this as a proof of concept and never use this technique without taking the necessary safety measures, and don't use it as a professional measure, as you might expose important network information.

We will start with an Arduino code, where we will have a LED connected to pin number 10 and the Arduino UNO permanently connected to the serial port.

The following is the Arduino code:

```
//#############################
//Author : Jorge Reyes Castro
//Arduino Home Security Book
//#############################
// Global variable
char messa;
int led = 10;
```

```
void setup() {
  Serial.begin(9600);               // set the baud rate
  pinMode(led, OUTPUT);             // Pin 10 = LED
  digitalWrite(led,HIGH);           // Initial Test ON/OF
  delay(500);
  digitalWrite(led,LOW);            //End of the Initial test
}

void loop() {
  if (Serial.available()>0){        // Serial port Open
                                    // Read Incoming message
    messa=Serial.read();
    if(messa=='F') {                // F = OFF
      digitalWrite(led, LOW);
    }
    if(messa=='N') {
      digitalWrite(led, HIGH);      // N = ON
    }
  }
}
```

The following is the python code for the `myweb.py` file:

```
import web
from time importsleep
import serial

port = serial.Serial('/dev/tty.usbmodemfd1311', 9600)   #Prepare Port
urls = (# url (browser) - name (class)
  '/', 'index',
  '/on', 'on',
  '/off', 'off',
  )

app = web.application(urls, globals(), True)

# Class index = main page
class index:
  def GET(self):
    return """<html>
      <head>
      <title>My First Home Security system</title>
      </head>
      <body>
      <body bgcolor="#1FB2FB">
      <h1 align="center"><font color="white">Command and Control</font>
</h1>
      <p align="center">Welcome to the graphical user interface, press the
button to turn off or turn on the LED.</p>
      <p align="center"><button  onclick="location.href='/on'">ON</button>
      <button onclick="location.href='/off'">OFF</button></p>
      <h4 align="center"><a href="/">Back</a></h4>
      </body>
      </html>"""

# Load when you click in "On"
```

```
class on:
  def GET(self):
      port.write("N");
#Send "N" through the Serial Port
      return """<html>
        <head>
        <title>My First Home Security system</title>
        </head>
          <body>
          <body bgcolor="#1FB2FB">
          <h1 align="center"><font color="white">Command and Control</font>
</h1>
          <h3 align="center">ON</h3>
          <h4 align="center"><a href="/">Back</a></h4>
          </body>
          </html>"""

# Load when you click in "Off"
class off:
  def GET(self):
      port.write("F");
#Send "F" through the Serial Port
      return """<html>
          <head>
          <title>My First Home Security system</title>
          </head>
          <body>
          <body bgcolor="#1FB2FB">
          <h1 align="center"><fontcolor="white">Command and Control</font>
</h1>
          <h3 align="center">OFF</h3>
          <h4 align="center"><a href="/">Back</a></h4>
          </body>
        </html>"""

if __name__ == "__main__":  #run app
  app.run()
```

Next, open a shell in the directory that you have the Python file in, and type:

```
Python myweb.py
```

The output looks like this:

```
http://0.0.0.0:8080/
```

Using a browser, go to `http://0.0.0.0:8080/`, and you should see the website. If you look at the shell that is open, you should be able to see something like this:

```
127.0.0.1:59217 - - [13/Apr/2015 01:41:18] "HTTP/1.1 GET /" - 200 OK
127.0.0.1:59217 - - [13/Apr/2015 01:41:19] "HTTP/1.1 GET /" - 200 OK
127.0.0.1:59217 - - [13/Apr/2015 01:41:21] "HTTP/1.1 GET /" - 200 OK
```

Summary

In this chapter, you have learned how to document your code, share it, keep track of its status, and maintain a backup. Finally, you created a simple graphical application that allows you to remotely control your Arduino UNO.

I urge you to and try to create a more complete website with more options, and when you have everything ready, use Git to share your creation.

This last point is crucial to our last chapter, in which we integrate everything we have learned so far and create the final project, so get ready.

Chapter 7. Interaction and Connectivity

This is the last chapter of the book. In this, we will create a system for detecting people to increase the capacity of our infrastructure. We give a brief introduction to the key elements for the course of this chapter. Finally, we will conclude with a practical example.

So far, we have always depended on a computer, whether desktop or laptop, to read the data obtained from our sensor. This was only for small local tests. To increase the power of our creation, we will use a new element in our scenario—a microcomputer.

I'm sure that you are aware of the several devices that can be used as a small, low-cost, and powerful computers with minimal power consumption. For our project, we will use the **Raspberry Pi**.

The Raspberry Pi

Raspberry Pi is a small computer board with USB ports (depending on model), an ARM processor (available in single-core and quad-core) that is widely used in mobile devices today. The ARM processor on the Pi is powerful, consumes low power, and is versatile). What makes the Pi so special is the **General Purpose Input/Output** (**GPIO**) port headers, which allows you to connect different components like ready-made shields and even basic electronic elements to it directly. With a very low price of about $35, it is very affordable. Above all, there are numerous Linux distributions based on what gives us great flexibility and *personalization* of the environment, you can install only the packages you need (those already familiar with Linux environment will feel at home with the Pi's UI interfaces). It lets you have full control over the services that are running, and utilizing the hardware to the maximum.

Note

To know more about ARM, visit:

- http://www.arm.com/products/processors/
- http://en.wikipedia.org/wiki/ARM_architecture

To know more about Raspberry Pi, visit:

- http://en.wikipedia.org/wiki/Raspberry_Pi
- http://en.wikipedia.org/wiki/General-purpose_input/output
- https://www.raspberrypi.org/
- https://www.raspberrypi.org/products/

There are numerous distributions for the Pi, we will use **Raspbian** for its great support and stability. It is based on the great **Debian** operating system, and ease of installation packages and libraries, thanks to manager **Advanced Package Tool** (**APT**).

Note

APT is a powerful command. I advise you to carefully read the information at http://en.wikipedia.org/wiki/Advanced_Packaging_Tool to better understand its use.

Visit http://elinux.org/RPi_Distributions if you want to know more about the operating system distributions that is supported by the Raspberry Pi. They are free to download, and you can choose the one you like. All are grades in its way, as it has very specific purposes (web servers, security tools, or even to play).

I do not pretend to be a comprehensive guide on how to install and configure the operating system for these devices, so quality material will be linked, which guides you through this simple process. I highly recommend using the NOOBS Installation Manager, which helps us in using more than one distribution. It will allow us to easily install any supported distribution.

Note

For more information on Raspbian & NOOB distribution, visit:

- https://www.raspberrypi.org/downloads/
- https://www.raspberrypi.org/help/noobs-setup/
- http://www.raspbian.org/

Once we have installed the distribution using the SD card, we proceed with the first ever boot of the Pi. It will be necessary to enable SSH connections (please change the default password, since any person can access sensitive information with the default password).

Setting up

Once you have burned the image on the SD card, you then connect the Pi, and with the help of a monitor and keyboard, perform a basic configuration. Once you are at the console, enter the following command line:

```
GeorgeRaspbian:~$ sudo raspi-config
```

After you enter the command you should get a GUI utility screen. Select the **Advanced** option and enable the SSH capability. Exit the utility and enter the following command to configure your new password:

```
GeorgeRaspbian:~$ passwd pi
```

Once we complete the preceding two steps, we can connect the Raspberry Pi to a router. To knowing the IP address of the Raspberry Pi you can use a smartphone or a computer with an application that scans your network and shows the connected devices and their IP addresses.

Note

Fingerbox is an application software that can scan the network to show the devices connected to the same network. To know more about this software, visit:

- http://www.overlooksoft.com/download
- http://www.overlooksoft.com/docs/Fingbox-User-Manual.pdf

Once we know the IP address (for example, 123.456.7.8), we proceed to connect to the Pi over SSH. Once connected we can now update the installed tools, and finally begin to prepare the environment of our project. Follow the given steps to connect remotely over SSH:

1. Using your PC/Smartphone/Mac, enter the following command to connect to the Pi remotely, for our project, making sure that you enter the Pi's IP address:

   ```
   ssh -X pi@123.456.7.8
   ```

2. Enter `-X` to force graphical user interface.
3. Next, it will request the password of the machine you are connecting to. While entering the password, you will not see anything that you type; this is normal. Enter the password and press the *Enter* key:
4. Enter the following command to update the list of the software repository for the Pi:

   ```
   sudo apt-get update
   ```

5. Once the list is updated, its time to update the Pi, using the renewed repository list:

   ```
   sudo  apt-get upgrade
   ```

6. Update the OS distribution:

   ```
   sudo apt-get dist-upgrade
   ```

7. Install the build-essential, cmake, and pkg-config application:

```
sudo apt-get install build-essential cmake pkg-config
```

we proceed to update to force-ensure that the reader is using the latest versions of all libraries, because the project needs to develop numerous libraries updated.

SSH updates are faster and more frequently when it connects to remote machines. (Many distribution managers do not have graphical installer packages.)

The preceding steps might take a while, so let's leave the Pi to get updated. Once it is ready, we can continue with the next steps. For now, let's learn about a new type of sensor that we need in our project.

Note

For more information on SSH, see the link. Also if you do not have an operating system with integrated SSH package by default, such as Linux and Macintosh OS may download the tools in the links provided:

- SSH: http://en.wikipedia.org/wiki/Secure_Shell
- Windows SSH client: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
- Android SSH Client: https://juicessh.com/
- IOS: https://itunes.apple.com/es/app/serverauditor-ssh-shell-console/id549039908?mt=8

Camera and IP Camera

We have a wide range and types of digital cameras, depending on the technology used, but a basic classification can be the following:

- **Wireless**: They don´t need wires, and can transmit the image through the air. They are very quick to deploy and relatively are expensive.
- **Wired**: Here, communication is performed by a wired link, which allows faster data transfer and also eliminates interference.

Cameras also differ in the type of sensor they use:

- **Specific**: It is able to capture very specific light spectra (infrared), and record images in slow motion or high definition.
- **Nonspecific**: They are general purpose cameras, that offers generic camera functionality for example, web cameras and simple digital portable cameras. Such cameras are inexpensive.

Wireless cameras can connect to a router or a network and you can access them from your computer’s browser or from your smartphone. The main problem is the security of the connection (if not configured properly, it can cause a real threat to privacy). Wireless cameras can even be placed on moving objects for more interesting images and footage (an example of this is an IP camera connected to a drone or a remote controlled quad-copter). You can make use of any old smartphone that you might have as an IP camera too. You will need to install third-party software that will allow us to connect to the device within the same Wi-Fi network.

There are several ways to create a mobile IP camera, you use such specific cameras, like as action sports cameras, or you can use your old unused smartphone (android phones generally offer more flexibility in this case) to output images inside a wireless network. I recommend searching the internet to find out more about how you can implement this. Visit http://www.howtogeek.com/139373/how-to-turn-an-old-android-phone-into-a-networked-security-camera/ for more information.

On the other hand, we have USB cameras that are very common and are found in every home. With these camera and the USB connection with certain adjustments, we can connect it to our RPI. Here’s a demonstration:

1. You must to connect the camera to the Raspberry Pi before you boot it up (`ssh -X option`)
2. To download the drivers for controlling the camera, enter the following command in the Raspbian terminal console:

   ```
   sudo apt-get install fswebcam
   ```

3. To download an image viewer, enter the following command:

   ```
   sudo apt-get install links2
   ```

4. Once you have the software installed, lets test and check if the camera can take

pictures. Use the following command:

```
cd /tmp/
fswebcam helloPi.jpg
```

As it is a test, we saved the images to a temporary directory. The test images taken will be deleted after restarting the machine.

5. If you have a blank or a blacked out image file, then enter the following command:

   ```
   fswebcam –p YUYV helloPi.jpg
   ```

6. To see the picture you just took, enter the following command:

   ```
   links2 –g helloPi.jpg
   ```

7. If you want to send the image in an email, follow the given steps:
   1. To install the SMTP server, enter the following command:

      ```
      sudo apt-get install ssmtp      # server to send
      ```

   2. Install the email utility tool:

      ```
      sudo apt-get install mailutils
      sudo apt-get install mpack       # attachment
      ```

   3. Enter the following in the `ssmtp.conf` file to configure our email address and the data messaging server:

      ```
      sudo nano /etc/ssmtp/ssmtp.conf
            AuthUser=user@gmail.com
            AuthPass=userpass
            mailhub=smtp.gmail.com:587
            UseSTARTTLS=YES
      ```

   4. Finally to send the email with the image as an attachment, enter this command:

      ```
      mpack –s "Hello" -D ./helloPi.jpg email@friend.com
      ```

It may seem silly to do it with each command, but we are now able to use these commands in a single script to send an email with a picture every time someone initiates a trigger action (for example, trying to unlock an NFC activated door more than three times). With such a facility, you can track and flag people with unauthorized access who may try to open the door.

As mentioned in the earlier chapters, we just do not want to just take a picture, we want to be able to process data captured by our camera and take decisions based on the results (for example, detecting the presence of authorized people).

OpenCV

To achieve image processing, we use the famous **OpenCV** library that is free to use and covers numerous languages such as C, Java, or Python. It is available and works with devices like smartphones, tables, and computers.

We will use the existing adaptation and implement these in Python.

Note

Caution: Implementing this project takes a lot of time, this depends on the version of the hardware, and there are steps in it that can take more than 10 hours to complete! Carefully follow the steps listed here in the correct order.

It may although seem like a very long process, but note that we will be compiling an extensive code on a **System on Chip** (**SoC**) processor, normally found in mobile or tables. At one time, it was common to have to wait many hours to compile heavy programs). I assure you that the result will be satisfactory.

For more information on OpenCV, visit the following links:

- http://opencv.org/
- http://en.wikipedia.org/wiki/OpenCV

If you have not yet updated the Pi with the latest upgrades, then I suggest that you install the updates before you start with the project. Many dependencies are also needed, which are beyond the scope of this book, but please make sure you have everything that is necessary installed.

Installing the application and its dependencies

Lets begin with the installation, follow the given steps in the order given:

1. Install pip and setuptools:

   ```
   cd /tmp/
   sudo apt-get install python2.7-dev
   wget https://bootstrap.pypa.io/get-pip.py
   sudo python get-pip.py
   pi@raspberrypi /tmp $ pip install -U setuptools
   pi@raspberrypi /tmp $ pip install -U pip
   ```

2. Install the math library:

   ```
   pip install numpy && echo "End of numpy" | mail -s "Numpy Install"
   yourmail@email.com
   ```

   You might have to wait an hour for the installation to finish; you can leave your computer and maybe have a cup of coffee.

3. Install the **virtualenv** to create an isolated Python environment:

   ```
   sudo pip install virtualenv virtualenvwrapper
   export WORKON_HOME=$HOME/.virtualenvs
   source /usr/local/bin/virtualenvwrapper.sh
   source ~/.profile
   mkvirtualenv cv
   ```

   Now we have a virtual environment `cv` (we can call as `$ workon cv`).

4. Install the dependencies that are needed to run our project:

   ```
   sudo apt-get install libjpeg8-dev libtiff4-dev libjasper-dev libpng12-
   dev libgtk2.0-dev libavcodec-dev libavformat-dev libswscale-dev libv4l-
   dev libatlas-base-dev gfortran
   ```

5. Download and install OpenCV by entering the following commands, this process will take about 10 hours to complete:

   ```
   cd..  # to switch back to the Home directory
   wget -O opencv-2.4.10.zip
   http://sourceforge.net/projects/opencvlibrary/files/opencv-
   unix/2.4.10/opencv-2.4.10.zip/download
   ```

6. Once downloaded, you need to extract the file before you can use it:

   ```
   unzip opencv-2.4.10.zip
   cd opencv-2.4.10
   ```

7. Now create a directory `build` and switch to the `build` folder:

   ```
   mkdir build
   cd build
   ```

   ## Note

   From this point on, it is very important not to make mistakes while typing the commands into the console, as these are long processes that can take as long as 10 hours; failure can occur even after many hours of work.

8. Make and install OpenCV:

   ```
   sudo cmake -D CMAKE_BUILD_TYPE=RELEASE -D
   CMAKE_INSTALL_PREFIX=/usr/local -D BUILD_NEW_PYTHON_SUPPORT=ON -D
   INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D
   BUILD_EXAMPLES=ON .. && echo "End of cmake" | mail -s "Cmake ok"
   yourmail@email.com
   ```

   ### Note

   No need to leave your computer ON (you can close the SSH connection), but you can not turn off the RPi, it still working!.

   ```
   sudo make    && echo "Make" | mail -s "Make Ok" yourmail@email.com
   sudo make install && echo "Make install" | mail -s "Make install"
   yourmail@email.com
   sudo ldconfig
   ```

9. Next, we configure the virtual environment:

   ```
   cd ~/.virtualenvs/cv/lib/python2.7/site-packages/
   ln -s /usr/local/lib/python2.7/site-packages/cv2.so cv2.so
   ln -s /usr/local/lib/python2.7/site-packages/cv.py cv.py#END
   ```

   At this point, we will have everything ready to start working on the project. I highly recommend that you to make a backup of the image installed on the Pi before you proceed any further. A backup will ensure that we do not lose all the work done if something goes wrong later.

   We will proceed to use one of the examples that come with the library, and also see the power of this technology. Please be patient, the first time you start one of these script may take a few minutes.

10. Connect to your Pi using the `ssh –X` option (it will allow us to open a remote session), making sure that you have the USB camera already connected to the Pi.
11. Now load the virtual environment using the following command:

    ```
    workon cv
    ```

12. Navigate and go to the `samples` folder under the OpenCV installed location:

    ```
    cd opencv-2.4.10/samples/python2/
    ```

13. Finally, now execute the `facedetect.py` script:

    ```
    ~/opencv-2.4.10/samples/python2 $ python facedetect.py
    ```

After a while, we can see how the program starts to detect your face, distinguishing the area of the eyes, and even the shape of the head. Incredible, right? This is truly amazing on what we have done with this tiny device.

We are now able to detect people and even identify them. I recommend that you stop here and experiment more on the lines we have just achieved. I urge you to try out other examples that are present online. Among other things, we can identify faces in the photograph, videos, even detect movement, or even superimpose a layer of augmented reality with the help of picture elements (think about how you can replace the rectangle that marks his eyes for another type of item, such as glasses).

Face detection

Right now, we have two options. We can either integrate our face recognition security system as it is in the example, or we can create a proprietary system that identifies people silently and stores the captured data directly to storage. This ensures that whenever the door opens for you, you can check and identify the people who have entered before you, or maybe even show us the persons face along with the time of entry. Here are the steps to create a face recognition program using Python:

1. Initiate a session on OpenCV using the following command:

   ```
   workon cv
   ```

2. Create a directory to store the images that will later be used in the Web interface:

   ```
   mkdir static
   ```

3. Now go into the newly created directory:

   ```
   cd static
   ```

4. Create a `face.py` file for the script:

   ```
   nano face.py
   ```

5. Type the following code into the file, or you can use any text editor to paste the code into the file:

   ```
   #!/usr/bin/env python
   ###################################
   #
   # Author : Jorge Reyes Castro
   # Arduino Home Security Book
   # Name: face.py
   ###################################
   import numpy as np
   import cv2
   import cv2.cv as cv
   from cv2 import *
   from video import *
   from common import clock, draw_str
   from time import gmtime, strftime

   ###################################
   #    FaceDetection
   ###################################

   # Optional we can insert the send email code (previous chapter)
   # Or can use the SerialPort code to control our Arduino

   def detect(img, cascade):
     rects = cascade.detectMultiScale(img, scaleFactor=1.3,
   minNeighbors=4, minSize=(30, 30), flags = cv.CV_HAAR_SCALE_IMAGE)
     # No face found
     if len(rects) == 0:
   ```

```
    print "NO FACE"
    return []
  rects[:,2:] += rects[:,:2]
# Optional, Save and print the current date
#  date = strftime("%Y-%m-%d %H:%M:%S", gmtime())
#  print "date" + " " + "Face Found"
#  imwrite("date" + "face.jpg", img)

  print "FACE FOUND"
  imwrite("face.jpg", img)
  return rects

###################################
#    MAIN
###################################

if __name__ == '__main__':
  import sys, getopt

  args, video_src = getopt.getopt(sys.argv[1:], '', ['cascade=',
'nested-cascade='])
  try: video_src = video_src[0]
  except: video_src = 0
  args = dict(args)
  cascade_fn = args.get('--cascade',
"../../data/haarcascades/haarcascade_frontalface_alt.xml")
  nested_fn = args.get('--nested-cascade',
"../../data/haarcascades/haarcascade_eye.xml")

  cascade = cv2.CascadeClassifier(cascade_fn)
  nested = cv2.CascadeClassifier(nested_fn)

  cam = create_capture(video_src)
  # Execute allways
  while True:
    # Prepare the camera
    ret, img = cam.read()

# Use the GrayScale gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    gray = cv2.equalizeHist(gray)

    t = clock()
    rects = detect(gray, cascade)
    dt = clock() - t
    if 0xFF & cv2.waitKey(5) == 27:
      break

###################################
#    END
###################################
```

6. Enter the following command to run the script file:

   ```
   ~/static $ python face.py
   ```

   You should see a similar output log on the screen: (Aim the camera at your face from at a distance to take a proper full-face image clearly):

```
NO FACE
NO FACE
NO FACE
NO FACE
NO FACE
FACE FOUND
NO FACE
```

7. After the program detects a face, it will save the image named as `face.jpg` in the `static` folder.

Note

Lets not dwell more in the concepts of facial recognition, as it requires extensive knowledge in this technology to understand it fully. It also requires an extensive documentation that explains the *ins* and *outs* of it.

At this point, you will know how to use this system. You can also associate multiple levels of safety along with the facial recognition, like not opening the door for a person if he/she is not authorized (for example, strangers whom you don't even know about). You can also even create a control interface that lets you see who the last person was to open the door (like checking the history tab in a browser). Please feel free to complete and perfect this code to add in more features.

C&C - Command and control

Finally, we will conclude by building a UI or Webpage to integrate this last element to our project. We can produce even greater result by making small changes in the code and UI elements. Follow the given steps to make a Web based GUI for your project:

1. Install the Pyserial and the webPy components on the Pi using the following command-lines:

   ```
   sudo pip install pyserialsudo pip install web.py
   ```

2. Create a script file `myweb.py`

   ```
   nano myweb.py
   ```

3. Enter the following code into the `myweb.py` file (make sure you copy the code from the code bundle file to ensure correct tab spaces as required the `.py` to correctly execute):

```
###################################
#
# Author : Jorge Reyes Castro
# Arduino Home Security Book
# Name: face.py
###################################
import web
from time import sleep
import serial

port = serial.Serial('/dev/ttyACM0', 9600)  #Prepare Port
urls = (                 # url (browser) - name (class)
 '/', 'index',
 '/on', 'on',
 '/off', 'off',
 )

app = web.application(urls, globals(), True)

# Class index = main page
class index:
  def GET(self):
    return """<html>
          <head>
          <title>My First Home Security system</title>
          </head>
          <body>
          <body bgcolor="#1FB2FB">
          <h1 align="center"><font color="white">Command and
Control</font></h1>

          <p align="center">Welcome to the graphical user interface,
press the button to turn off or turn on the LED.</p>

          <p><img src="/static/face.jpg" alt="HummanDetect"
```

```
align="center"></p>

          <p align="center"><button
onclick="location.href='/on'">ON</button>
          <button onclick="location.href='/off'">OFF</button></p>

          <h4 align="center"><a href="/">Back</a></h4>
          </body>
          </html>"""

# Load when you click in "On"
class on:
  def GET(self):
      port.write("N");  #Send "N" through the Serial Port
      return """<html>
          <head>
          <title>My First Home Security system</title>
          </head>
          <body>
          <body bgcolor="#1FB2FB">

          <h1 align="center"><font color="white">Command and
Control</font></h1>

          <h3 align="center">ON</h3>
          <h4 align="center"><a href="/">Back</a></h4>
          </body>
         </html>"""

# Load when you click in "Off"
class off:
  def GET(self):
      port.write("F");    #Send "F" through the Serial Port
      return """<html>
          <head>
          <title>My First Home Security system</title>
          </head>
          <body>
          <body bgcolor="#1FB2FB">
          <h1 align="center"><font color="white">Command and
Control</font></h1>

          <h3 align="center">OFF</h3>
          <h4 align="center"><a href="/">Back</a></h4>
          </body>
        </html>"""


if __name__ == "__main__": #run app
  app.run()

###################################
#    END
###################################
```

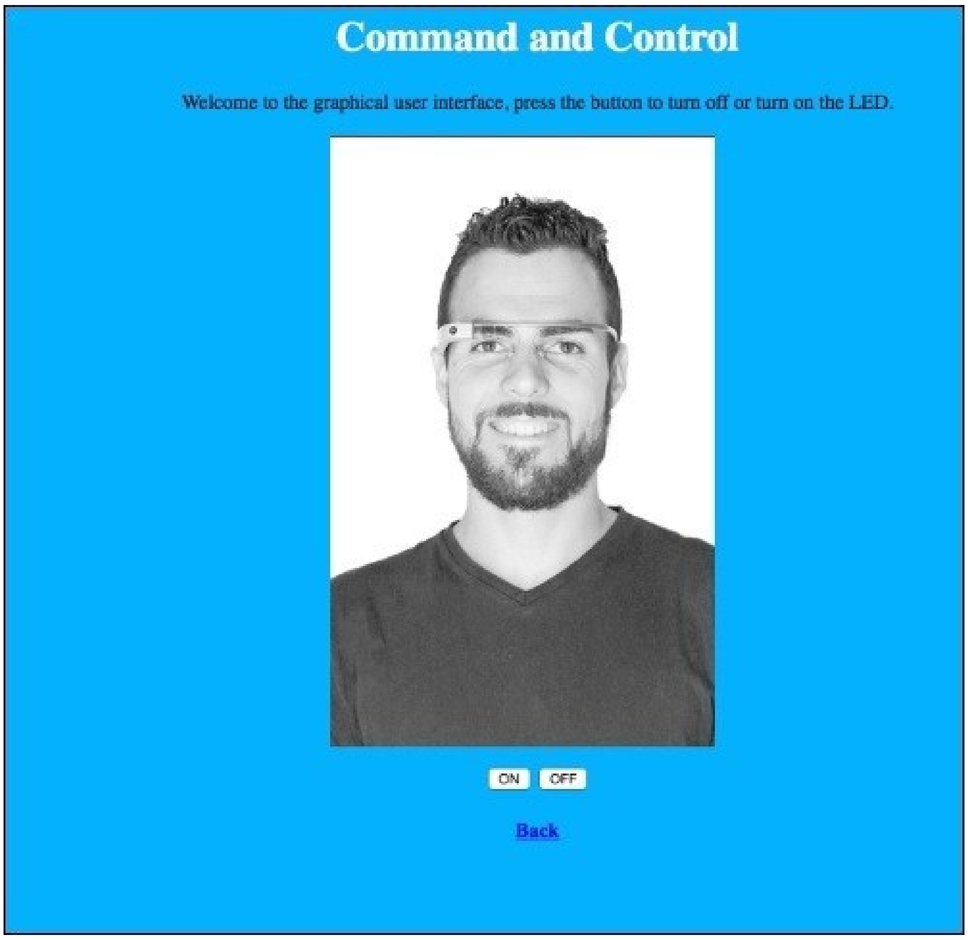
4. Go to the `static` folder:

```
cd static
```

5. Now execute the `face.py` and the `myweb.py` scripts:

```
python face.py
python ~/myweb.py
```

6. Point your face at the camera and refresh the Webpage, and you should see the face of the detected person appear on screen, shown as follows:



Command and Control

Welcome to the graphical user interface, press the button to turn off or turn on the LED.

An example of the output

Summary

With this, we conclude our introduction to control systems and domestic surveillance after touring the various aspects of a project of this wingspan and learning how to deal with the complexities, designing the various modules, and improving them.

I appreciate your company until the end of this book, and I hope we meet again soon in the lines of another fantastic project. I have enjoyed writing and creating the projects in this book, and I hope you have enjoyed the experience too. This is the time to put into practice what we have learned by creating the project for our home or workplace and complete this learning cycle.

Index

X

- X10
  - URL / Wired and wireless security systems.