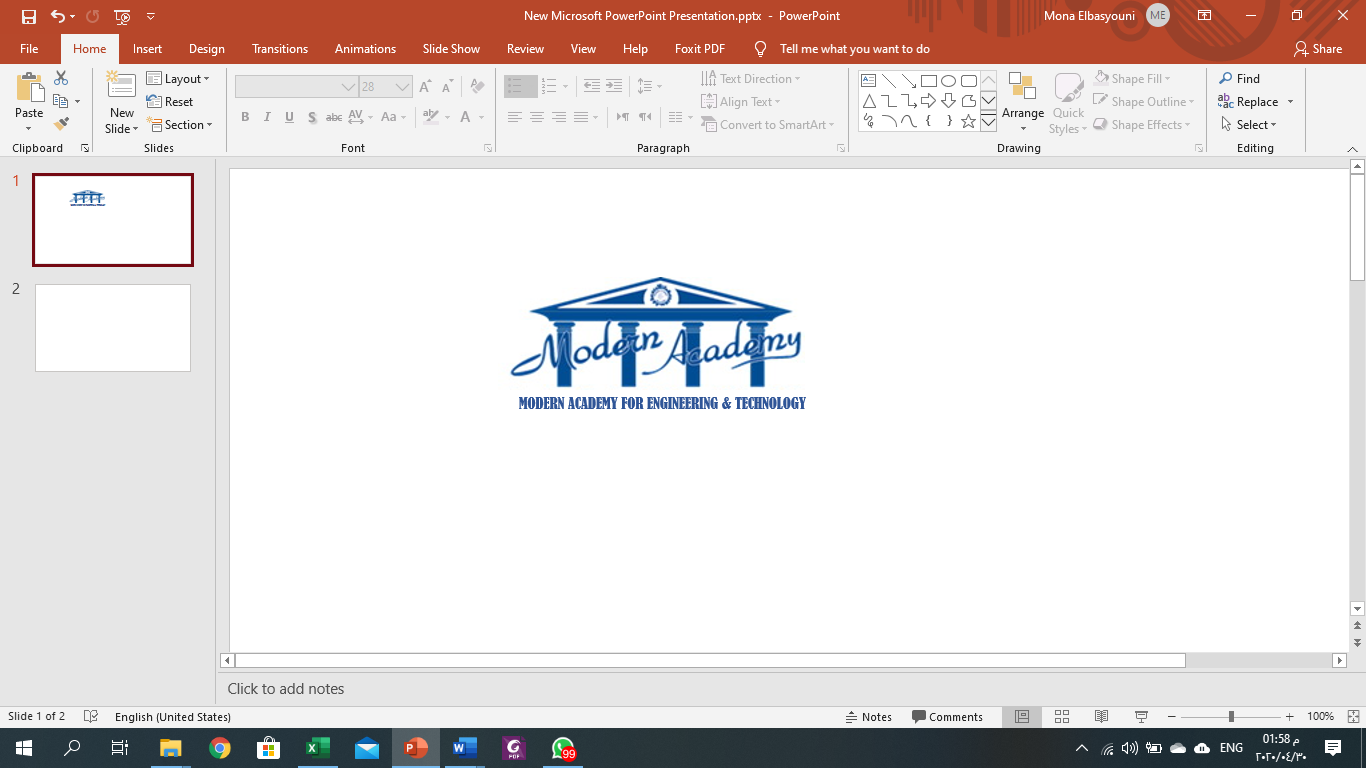
**MODERN ACADEMY**

**FOR ENGINEERING & TECHNOLOGY**

**Computer Department**

**Spring Semester - Academic Year 2019/2020**

****

**PROJECT name : IOT**

|  |
| --- |
| **Course Instructor** |
| **Prof. Dr. Wafae Boghdady** |

|  |  |  |
| --- | --- | --- |
| Section | الاسم | No. |
| 1 | احمد حسن محمد حسن صقر | 1 |
| 1 | حلمي شلقامي حلمي محمد | 2 |
| 1 | محمد ايمن محمد خليفه | 3 |
| 1 | حسام الدين مدحت حسين ابراهيم | 4 |
| 1 | أحمد عرفة فوزى عبد الله حسين | 5 |
| 1 | محمد علاء الدين مصطفي عبد الحميد جوده | 6 |
| 1 | امير عصام علي عثمان | 7 |

**Table of Contents**

[Abstract 6](#_Toc42549114)

[List of Figures 7](#_Toc42549115)

[List of Abbreviation 10](#_Toc42549116)

[CHAPTER 1: Introduction 12](#_Toc42549117)

[IOT IMPLEMENTATION PROJECTS 12](#_Toc42549118)

[Sensors 13](#_Toc42549119)

[Local processing and storage devices 13](#_Toc42549120)

[Network and Internet 13](#_Toc42549121)

[Cloud 14](#_Toc42549122)

[Hardware 14](#_Toc42549123)

[Microcontrollers 14](#_Toc42549124)

[Computers 15](#_Toc42549125)

[Development devices 16](#_Toc42549126)

[Software 17](#_Toc42549127)

[Sensors and peripherals 18](#_Toc42549128)

[Sensors 18](#_Toc42549129)

[Button 20](#_Toc42549130)

[Button Denounce 20](#_Toc42549131)

[Potentiometer 21](#_Toc42549132)

[Thermistor 21](#_Toc42549133)

[Temperature sensor 23](#_Toc42549134)

[Light Sensor 23](#_Toc42549135)

[Gas Sensor 24](#_Toc42549136)

[Distance Sensor 24](#_Toc42549137)

[Digital sensors 24](#_Toc42549138)

[SPI 24](#_Toc42549139)

[I2C 25](#_Toc42549140)

[LED 27](#_Toc42549141)

[7 segment display 27](#_Toc42549142)

[Shift register 29](#_Toc42549143)

[LCD 31](#_Toc42549144)

[What's a web server and HTTP? 31](#_Toc42549145)

[Https has two important security roles. 32](#_Toc42549146)

[Webservers on gadgets using Wyliodrin 33](#_Toc42549147)

[Web services 36](#_Toc42549148)

[JQuery 37](#_Toc42549149)

[Web sockets 37](#_Toc42549150)

[AngularJS 37](#_Toc42549151)

[CHAPTER 2: Used Martials and its Features 39](#_Toc42549152)

[2.1 RaspberryPi 39](#_Toc42549153)

[Features: 40](#_Toc42549154)

[GPIO 40](#_Toc42549155)

[Voltages 41](#_Toc42549156)

[On the board are two 5V pins and two 3V3 pins, as well as several ground pins (0V), which are unconfigurable. The remaining pins are all 3V3 pins with general purpose, meaning outputs are set to 3V3, and inputs are sensitive to 3V3. 41](#_Toc42549157)

[Outputs 41](#_Toc42549158)

[They can be set to high (3V3) or low (0V) with a GPIO pin designated as an output pin. 41](#_Toc42549159)

[Inputs 41](#_Toc42549160)

[More about GPIO 42](#_Toc42549161)

[GPIO pinout 43](#_Toc42549162)

[2.2 H-bridge 43](#_Toc42549163)

[Features: 43](#_Toc42549164)

[2.3 Wild Thumper 45](#_Toc42549165)

[Specifications: 45](#_Toc42549166)

[CHAPTER 3: Code Explanation 46](#_Toc42549167)

[3.1 HTTP Server 46](#_Toc42549168)

[3.2 RaspberryPi 50](#_Toc42549169)

[3.2.1 Pulse-width modulation (PWM) 50](#_Toc42549170)

[3.2.2 Motor header code 51](#_Toc42549171)

[3.2.3 Definition of Motor Function Code 53](#_Toc42549172)

[3.2.4 The Main Project Code 58](#_Toc42549173)

[CHAPTER 4: Simulation 60](#_Toc42549174)

[CHAPTER 5: Conclusions and Future Work 61](#_Toc42549175)

[Conclusions 61](#_Toc42549176)

[Future Work 61](#_Toc42549177)

[CHAPTER 6: References and Appendices 62](#_Toc42549178)

# Abstract

In this project we are building a web-controlled surveillance robotic using raspberry pi and a webcam. This could be a valuable and modest security and spy device, which have numerous configurable alternatives. In this IOT Project, we are basically utilizing Raspberry Pi, camera module, 4 haggles cameras with robot body to fabricate this Robotic vehicle. The primary goal of this project is to build up a virtual environment for recognizing suspicious and targeted places for client with no loss of human life. It depends on development of a robot vehicle for observing/spying the suspicious objects. It can ceaselessly monitor the objects. Robot can move in all directions (left, right, forward and backward). It is utilized for video surveillance and remotely control the specific spot utilizing Wi-Fi as medium. The webcam which is put on the robotic unit will capture the video and it transmits lively to the remote end.

# List of Figures

[Figure 1 IOT 9](#_Toc42552529)

[Figure 2 Sensors 10](#_Toc42552530)

[Figure 3 Microcontroller Kit. 12](#_Toc42552531)

[Figure 4 Raspberry Pi 12](#_Toc42552532)

[Figure 5 ChipKIT 13](#_Toc42552533)

[Figure 6 LounchPad 13](#_Toc42552534)

[Figure 7 spark core wifi board 14](#_Toc42552535)

[Figure 8 nucleo stm32f4 14](#_Toc42552536)

[Figure 9 embedded linux boards 15](#_Toc42552537)

[Figure 10 Wyliodrin Studio 16](#_Toc42552538)

[Figure 11 Button 18](#_Toc42552539)

[Figure 12 Button Denounce 19](#_Toc42552540)

[Figure 13 Thermistor connection 20](#_Toc42552541)

[Figure 14 Thermistor in programing 20](#_Toc42552542)

[Figure 15 Light Sensor connection 21](#_Toc42552543)

[Figure 16 Light Sensor in programming 22](#_Toc42552544)

[Figure 17 SPI 23](#_Toc42552545)

[Figure 18 I2C 24](#_Toc42552546)

[Figure 19 connection 24](#_Toc42552547)

[Figure 20 I2C in programing 25](#_Toc42552548)

[Figure 21 7 segment display connection 26](#_Toc42552549)

[Figure 22 7 segment display in programing 27](#_Toc42552550)

[Figure 23 Shift register connection 28](#_Toc42552551)

[Figure 24 Shift register in programing 28](#_Toc42552552)

[Figure 25 LCD connection 29](#_Toc42552553)

[Figure 26 LCD in programing 29](#_Toc42552554)

[Figure 27 HTTP 30](#_Toc42552555)

[Figure 28 Webservers using Wyliodrin 32](#_Toc42552556)

[Figure 29 Webservers response 32](#_Toc42552557)

[Figure 30 Webservers response using Wyliodrin 33](#_Toc42552558)

[Figure 31 Webservers response files 33](#_Toc42552559)

[Figure 32 edit web template node 34](#_Toc42552560)

[Figure 33 edit web template node files 34](#_Toc42552561)

[Figure 34 AngularJS files 36](#_Toc42552562)

[Figure 35 AngularJS files and web response 36](#_Toc42552563)

[Figure 36 AngularJS in code 36](#_Toc42552564)

[Figure 37 Raspberry Pi 3 37](#_Toc42552565)

[Figure 38 Raspberry Pi Pin Numbers 38](#_Toc42552566)

[Figure 39 Raspberry Pi Pin Assignment 39](#_Toc42552567)

[Figure 40 BTS7960B H-bridge 41](#_Toc42552568)

[Figure 41 Wild Thumper 43](#_Toc42552569)

[Figure 42 Java code: import section 44](#_Toc42552570)

[Figure 43 Java code: Create new HTTP\_Server 45](#_Toc42552571)

[Figure 44 Java code: HTTP\_Server Function 45](#_Toc42552572)

[Figure 45 Java code: Server Start 46](#_Toc42552573)

[Figure 46 Java code: Textbox Creation 46](#_Toc42552574)

[Figure 47 Java code: Key Listener 47](#_Toc42552575)

[Figure 48 Java code: response status code 47](#_Toc42552576)

[Figure 49 Java code: send reply 48](#_Toc42552577)

[Figure 50 Ideal pulse-width modulation (PWM) 49](#_Toc42552578)

[Figure 51 Motor header code: Public section 50](#_Toc42552579)

[Figure 52 Motor header code: Private section 51](#_Toc42552580)

[Figure 53 Definition Motor Code: motor function 52](#_Toc42552581)

[Figure 54 Definition Motor Code: forward function 53](#_Toc42552582)

[Figure 55 Forward Motion 53](#_Toc42552583)

[Figure 56 Definition Motor Code: backward function 53](#_Toc42552584)

[Figure 57 Backward Motion 53](#_Toc42552585)

[Figure 58 Definition Motor Code: turn Right function 53](#_Toc42552586)

[Figure 59 Turning Right Motion 53](#_Toc42552587)

[Figure 60 Definition Motor Code: turn Left function 54](#_Toc42552588)

[Figure 61 Turning Left Motion 54](#_Toc42552589)

[Figure 62 Definition Motor Code: init function 54](#_Toc42552590)

[Figure 63 Definition Motor Code: Right side functions 55](#_Toc42552591)

[Figure 64 Definition Motor Code: Left side functions 55](#_Toc42552592)

[Figure 65 Definition Motor Code: stop functions 55](#_Toc42552593)

[Figure 66 Definition Motor Code: stop functions 56](#_Toc42552594)

[Figure 67 Definition Motor Code: stop functions 56](#_Toc42552595)

[Figure 68 Main Project code: initialization 57](#_Toc42552596)

[Figure 69 Main Project code: Get response status code 58](#_Toc42552597)

[Figure 70 Main Project code: Condition block 59](#_Toc42552598)

[Figure 71 Simulation 59](#_Toc42552599)

# List of Abbreviation

LAN ---- Local Area Network

UK ---- United Kingdom

CPU ---- Central Processing Unit

BLE ---- Bluetooth Low Energy

RAM ---- Random Access Memory

USB ---- Universal Serial Bus

HDMI ---- High-Definition Multimedia Interface

CSI ---- Camera Serial Interface

DSI ---- Display Serial Interface

SD ---- Serial Digital-Card

GPIO ---- General-Purpose Input/Output

PWM ---- Pulse-Width Modulation

SPI ---- Serial Perioheral Interface

MISO ---- Multiple Output, Single Input

SCLK ---- Serial Clock Signal

I2C ---- Inter-Integrated Circuit

EEPROM ---- Electrically Erasable Programmable Read-Only Memory

PCB ---- Printed Circuit Board

OS ---- Operation System

DC ---- Direct Current

HTTP ---- Hyper Text Transfer Protocol

MPPT ---- Maximum Power Point Tracking

ARM ---- Advance RISC Machine

HTML ---- Hypertext Markup Language

LED ---- Light Emitting Diodes

LCD ---- Liquid Crystal Display

ADC ---- Analog-to-Digital Converter

CO2 ---- Carbon Dioxide

GND ---- Ground

VCC ---- Voltage Common Collector

MOSI ---- Multiple Input, Single Output

SCL ---- Clock Line

SDA ---- Data Line

URL ---- Uniform Resource Locator

API ---- Application Programming Interface

# CHAPTER 1: Introduction

## IOT IMPLEMENTATION PROJECTS

This course is going to explain what IOT represents, what it consists of, also name some of the most important hardware which make it happen and try to give IOT a proper understanding of its meaning and importance. The Internet of things has become a very widely spread concept in the last few years. The reason for this is mainly the need to computerize and control most of the surrounding objects and have access to data in real time. Think about parking sensors, about phones which can check the weather and so on. The Internet of Things represents the whole way from collecting data, processing it, taking an action corresponding to the signification of this data to storing everything in the cloud. All this is made possible by the internet. Let's take sensors as an example, they collect data and send it to a processing device, which will perform the convenient actions. Then, the data will be stored locally and, by using the internet, it is subsequently sent out to the cloud. The problem here is that the data stored in the cloud is sometimes not useful. There is not enough local processing happening before data is saved in the cloud. The ideal scenario towards which the Internet of Things is headed, would be to have a computer store data locally, process it, check for abnormalities or search for relevant segments and upload only this information to the cloud. This concept is called either Edge, according to Intel, or fog computing as stated by Cisco, implying what happens before the cloud. The IOT is involved in medicine, agriculture or transportation due to sensors and cloud storage.

IOT Stack a basic Internet of Things system consists of the following components:



Figure 1 IOT

## Sensors

They transform analog data given from scanning the environment to digital data, but they merely do any processing. On the bright side, they don't consume much power and can live on batteries for a long time. Sensors are present in everyday life more than you would expect.

They improve industry, agriculture, homes, transportation or smart phones for example. They are tools which help monitoring the environment, collecting data about it and, with the help of computers, acting accordingly.



Figure 2 Sensors

## Local processing and storage devices

Local processing devices are the second level and third in IOT. At this point, data is locally stored and processed, ideally not sent forwards unless relevant. This part is explained in detail in the hardware section, as said devices are nothing more than microcontrollers and embedded boards, which handle the data they receive from the sensors.

## Network and Internet

There is hardware which connects to the previously described devices, pulls out data and sends it to the cloud to be stored. There are 4 protocols used at this level: CoAP, MQTT (less secure and designed for machine to machine communication), HTTP (web protocol) and XMPP which functions as a chat.

## Cloud

In the cloud, which comes next, data is collected and the main goal is for it to reach the point of making predictions based on the stored information. The cloud however, even though it represents one of the most useful features of the internet, is not used properly. Data sent to the cloud didn't reach the level of being formerly processed. Which means there is no preselected data. The cloud is constantly loaded with irrelevant information and thus losing its property of being practical.

## Hardware

An important subject concerning the IOT is open hardware. The designers of such hardware publish the schematics which, when the current started, was a real boost for the IOT.

## Microcontrollers

Microcontrollers were the first one to appear as an option for developers. They are small computing devices, easily connectable to hardware. The development tools were, however, an obstacle and made using them very complicated.

The first easy to use microcontroller was Arduino. It is a small programmable device, on which you can run a simple, open source software called firmware. The only fault was that it didn't have enough processing power and, as a result, no more than 4 network connections were supported. It’s RAM being of 2KB. Shortly, the Arduino does not run an Operating System, what it runs is called Firmware. Basically, it runs only one program. The result is that you can estimate what program sequence gets executed at a certain time. Program written to the Arduino remain there until they are replaced with another program. Even when powered off, Arduino stores

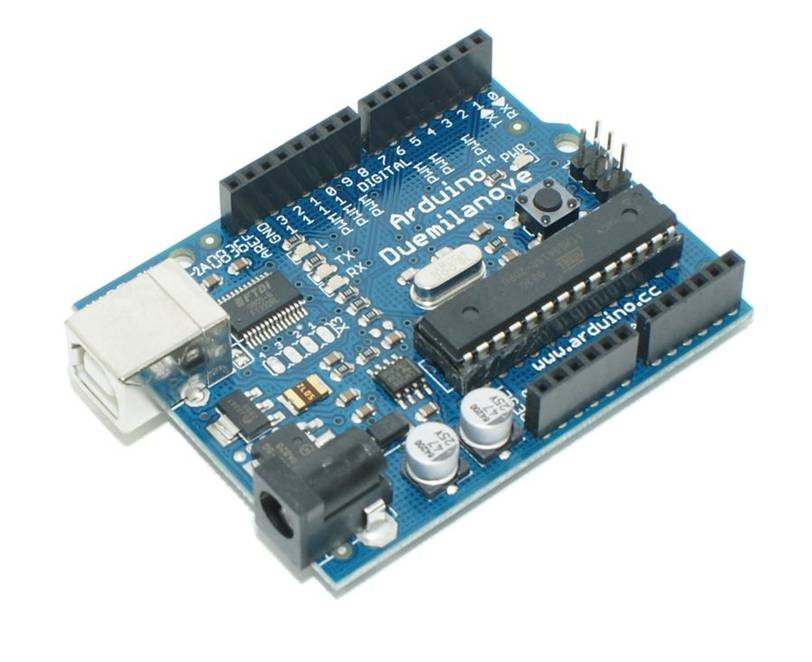


Figure 3 Microcontroller Kit.

**Its software:**

## Computers

Conversely, since it is a computer, the Raspberry Pi runs an Operating System. That means that you can run multiple programs on it and you can run applications that use Internet services. However, this also implies that the application you run on the Raspberry Pi is not real time, thus you cannot estimate when a certain sequence will get executed. Raspberry Pi came second with the qualities of being cheap and useful. It is actually a small computer, it runs Linux as operating system and has a full network system, solving thus the processing problem of the Arduino.



Figure 4 Raspberry Pi

## Development devices

The appropriate hardware can be found by checking many aspects. There are many boards, but they are good for different kinds of projects.

For sensor handling Arduino, ChipKIT and LounchPad are the best options.

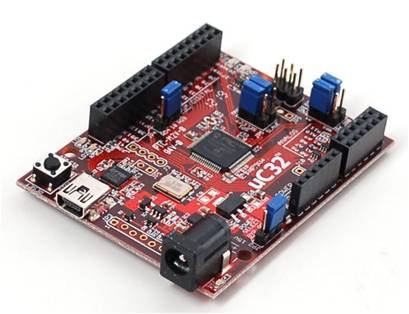


Figure 5 ChipKIT



Figure 6 LounchPad

For processing: the STM32 with 128KB of RAM, Particle with an ARM chip and Wi-Fi, Espino which is actually a JavaScript machine and so, you can write JavaScript code to run on it.

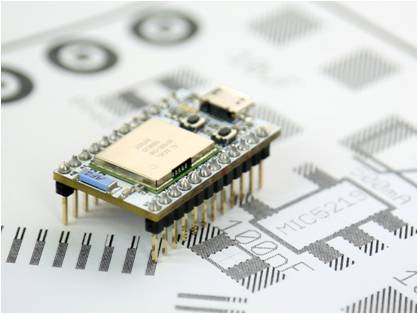


Figure 7 spark core wifi board

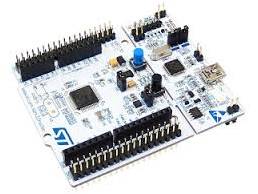


Figure 8 nucleo stm32f4

For processing and network, there are the Raspberry Pi; Intel Galileo which handles hardware pretty good, has 256MB of RAM and a Quark processor; The Intel Edison which comes as an improvement to the Galileo, can be used with Wi-Fi and Bluetooth and has a 4GB flash memory; Beagle bone Black also brings flash memory; UDOO Neo is a combination of a Raspberry Pi and an Arduino, project started on Kickstarter; Parallels which is very different from the others, being a prototyping board, plus, it has a chip which processes 16 or 64 programs at a time



Figure 9 embedded linux boards

## Software

Prototyping is a necessary process while building a professional product. The software used for it should be fast to write, easy to deploy when finished and it’s there to make a proof of concept. However, it is not supposed to be user level grade.

Prototyping has the only property of making a statement, whereas professional programming can’t have any faults, can’t break and has to meet the client’s needs. The software’s built for this job are Eclipse, VIM, MBED- online platform where you can write the code and download the binary file for the board-, Intel XDK which uses JavaScript as language for developing and also offers HTML as an alternative.

The field of data acquisition and analysis brings:

Lively, which collects data and displays it, offers libraries for you to integrate into your project, send the latter to the cloud. Here, you can add graphs and monitored everything.

Microsoft Azure, through which you push data to the cloud with failure detection as target. With this software, there’s no need for machine learning algorithms when you can use Microsoft’s technology and knowledge to build prediction models.

Solution builders such as Wyliodrin allow prototyping and build solution projects to be sent to the clients.

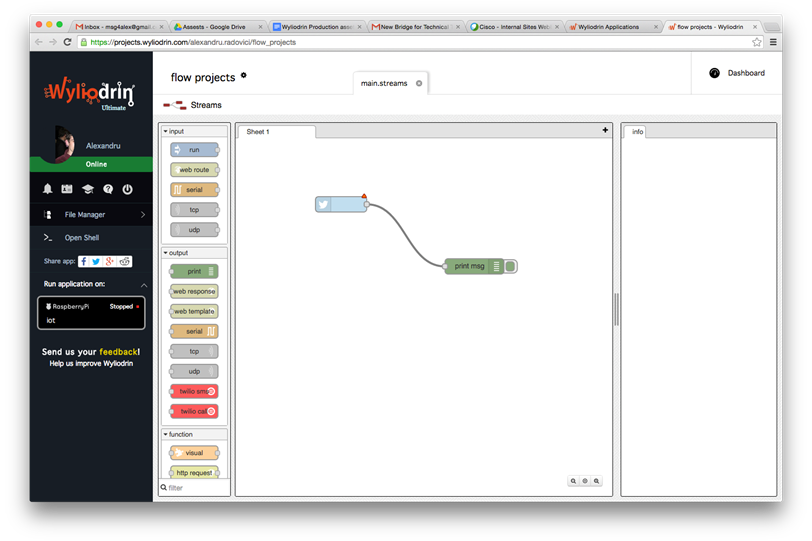


Figure 10 Wyliodrin Studio

## Sensors and peripherals

Slides are available from the link

This course will go through both analog and digital sensors, describing how they function and different ways of connecting and using them. More than this, it will present some other important peripherals, from the basic ones: LEDs to be more complicated - shift registers or significant ones- LCDs.

## Sensors

Sensors are devices that scan the environment and get data from it. Some examples of sensors are: thermistors, buttons (because they send a value, sensing the environment), photo resistors, infra-red sensors, distance sensors.

Types of sensors:

Analog Sensors - send an analog value that needs to be processed by the ADC

Digital Sensors - send a digital value

Basically, all the sensing parts are analog.

For the analog sensor the sensing part is directly connected to a micro controller. They are easy to interface; you only need an ADC to do some computing to find the real value.

Digital sensors are more complicated because the micro controller that processes the data is already in the box of the sensor and it sends the data out on a communication channel that implements some type of protocol. In order to get the data, the device needs to communicate with the sensor via the protocol.

For the analog sensors there are two types: with two or three pins.

An analog sensor will most probably be connected as a voltage divider. Such a sensor has a variable resistance, according to a specific environment element. Take a light sensor as an example: the more light there is, the lower the resistance. For the temperature sensor, the higher the temperature, the lower the resistance. You practically measure the voltage drop in the circuit. The 2 leg and 3 leg analog sensors are mentioned above. The sensors that have 3 pins usually have the voltage divider already integrated: one pin goes to VCC, one to the Ground and the third pin goes to the analog input. For the sensors withe 2 legs, you have to build the voltage divider yourself by adding a resistance to the circuit.

If you switch VCC with ground, some sensors will burn.

Another particularity of the sensors is that they will introduce some errors in the measurements, depending on the quality of the sensor. These parts usually increase in accuracy if they have the voltage divider inside as they bring some corrections. More to this, the sensors are not linear.

One simple way to connect errors in reading or the ones that occurred due to the fact that the circuit is not perfect is to average them (usually using 1000 samples). This works on micro controllers, but on computer boards, it might result in reading the same value over and over.

## Button

It is one of the simplest sensors that you can connect. It is in fact an analog sensor, but it can only report 2 values: either 1 or 0. If the button is released, the resistance is infinite, if it is pressed, the resistance becomes 0 and the input will be directly connected to the ground.

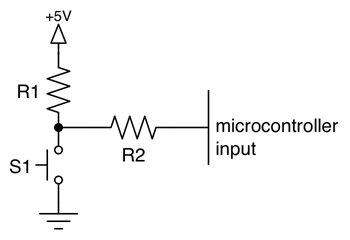


Figure 11 Button

The button becomes a pull-up resistor because when the button is not pressed it pulls up the voltage up to 5V. You need a really small resistor so that when the button is in the air the voltage drop will be insignificantly low, so that you can still read 5 volts. The problem comes up when you press the button for a longer period of time. In conclusion, you need a resistor sufficiently high that you won’t drain the source too fast, but sufficiently low to be able to read the value 1.

If R1 is replaced by S1 it will be called a pull-down resistor: it will pull down the voltage from that point to the ground the moment the button is not pressed. When the button is pressed it will stop the current from flowing too fast and having a short circuit. The input will be connected to VCC.

## Button Denounce

This problem is really visible on micro controllers because they sample fast. Buttons are imperfect, so from the moment you start pressing the button, there is a short time when the button’s connectors start approaching and the distance is really small resulting in an electric discharge. The resistance will be neither 0, nor infinite, so for a short period of time the resistance will vary.

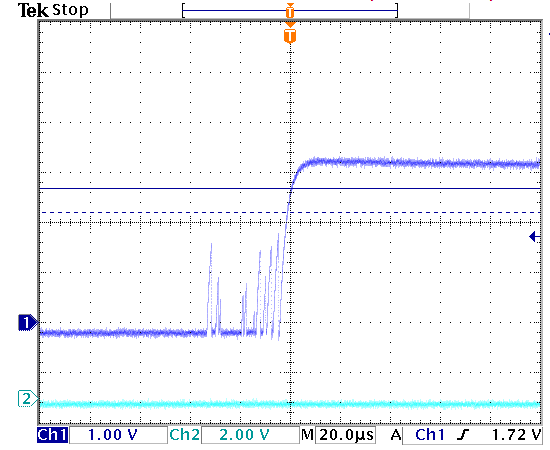


Figure 12 Button Denounce

With the microelectronics, you will see many values of 1 and 0 continuously changing. One way to denounce is to average the values and if the result is different from 0 or 1, then to discard the measurement because the button bounced. Another way is to use a trigger which, for an amount of time won't read any other value thus not detecting the edges. Expensive microcontrollers have denouncing circuits.

## Potentiometer

A potentiometer is a variable resistance, with 3 pins. Usually it is a full resistor and the pin is connected in the middle (it can float around the full resistor and split it into two resistors so it works as a voltage divider). You can connect it in two ways: either by making a voltage divider, choose R1 and connect two of the pins in the voltage divider (the middle one and one of the extremities), or by connecting one pin to VCC, one the Ground and the middle one to the input. To have a linear function as input from the sensor some computing is requested.

## Thermistor

Let's see an example of how an analog sensor can be connected to a board and how an application looks in Visual Programming for the thermistor, for instance. This sensor is still a variable resistor that changes its value with the temperature. You can compute the actual temperature using that value and a formula that takes into account the thermistor`s parameters, among which the resistance at 25 degrees. This kind of sensor has a low accuracy and is used to get a rough idea about the temperature.

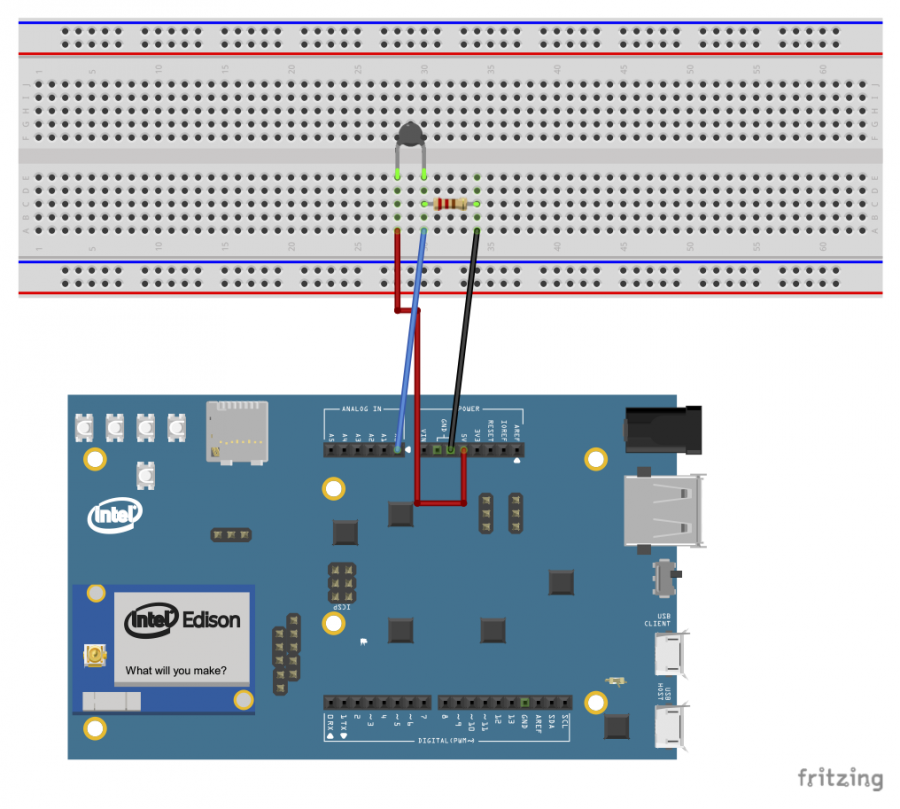


Figure 13 Thermistor connection

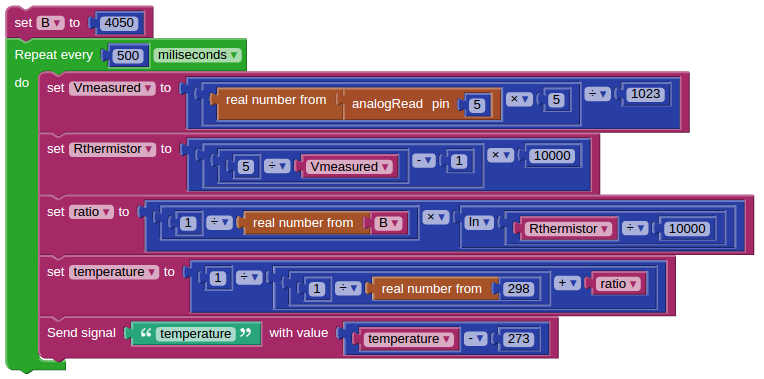


Figure 14 Thermistor in programing

## Temperature sensor

Note that there is a difference between the two sensors, consisting of the fact that the temperature sensor has a linear resistance-temperature characteristic, while the thermistor usually has the output voltage as a logarithmic or exponential function. Also the formulas to find the temperature in Celsius degrees are different.

## Light Sensor

This is also a resistance that changes its value due to the amount of photons it receives. Measuring the voltage drop on the resistance, you can get an idea about the light intensity. How does it actually work? If there is much light, there will be many photons in the photocell, thus less electrons to stop the electric current, which means the resistance of the sensor is lower.

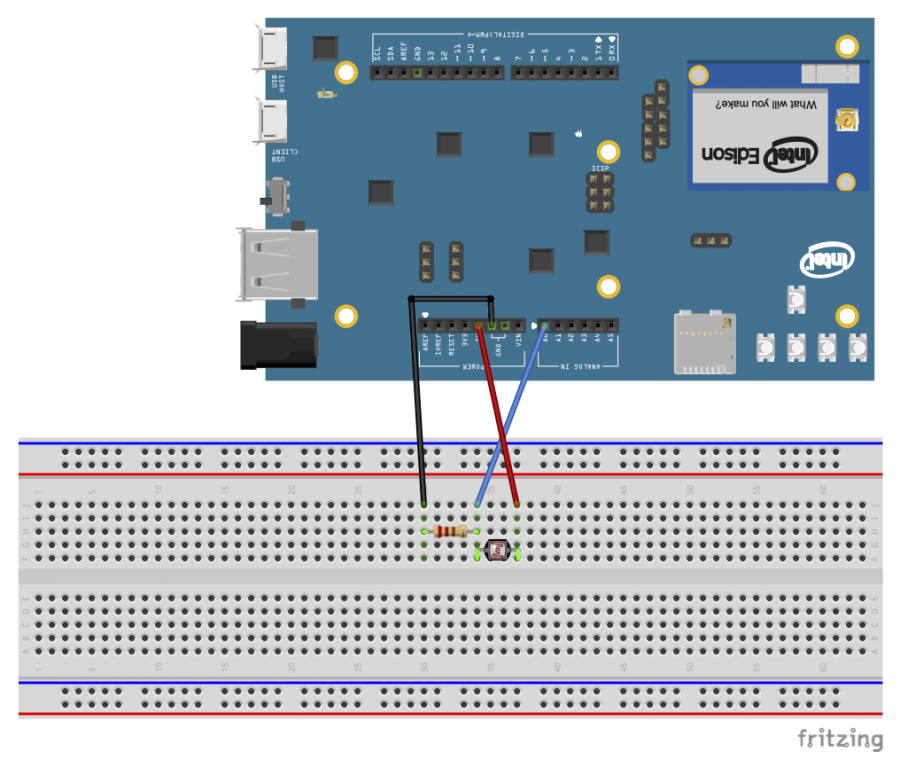


Figure 15 Light Sensor connection

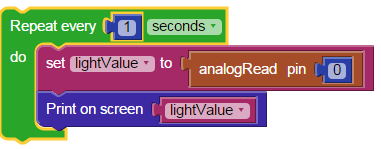


Figure 16 Light Sensor in programming

## Gas Sensor

This sensor detects CO2. Pay attention that in order to detect gas it uses an exothermic chemical reaction therefore it heats up so don`t touch it while it is connected to the board. More importantly pay attention that this sensor IS NOT for safety-important applications.

## Distance Sensor

This sensor works like a sonar. It has 4 pins. Two of them are the power pins: GND and VCC, and the other two are the trigger and the echo pins. The trigger will send out an ultrasonic signal and wait for the response. The moment the signal is sent out, the echo pin is 1, when the response comes, this pin turns to 0. The distance is measured by multiplying the time needed for the response to come with the speed of sound. Another type of sensor is an infrared one. It has an ADC inside so computing the distance is after a simple formula.

## Digital sensors

Digital sensors work with either of the two protocols: SPI or I2C.

## SPI

This protocol involves several slaves and a master. The communication is always initiated by the master. The pins for such a sensor are SS (slave select}, MISO, MOSI, Clock, each with their role. MOSI stands for master out, slave in, which means that a line is created to send data from the master to the slave. MOSI is the line that facilitates the communication in the opposite direction: the slave writes on the line and the master samples.

The master is the one who generated the clock. SS pin will be 0 when the slave is active and conversely 1 when the slave is inactive. In the first case, the slave waits for the clock to transfer data. One master has pins for each slave, but it can’t work with more than one slave at a time, since the MISO and MOSI lines would get impossible to use.

Communication in this protocol is always an exchange.

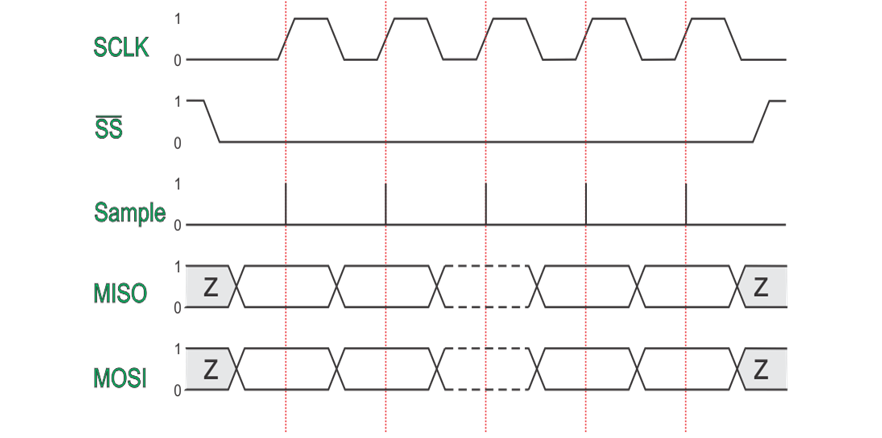


Figure 17 SPI

## I2C

This protocol works with one master and several slaves as well. This time there is no slave selection, but each slave has a fixed address.

It only uses two lines: SDA and SCL: serial data and serial clock line. This protocol is called a half-duplex as it only uses one line for communication. The transfer of data goes as follows. The master sends the address. The slave identifies itself, sends out and acknowledgement message, then the master writes or reads the data, all this on the same line.

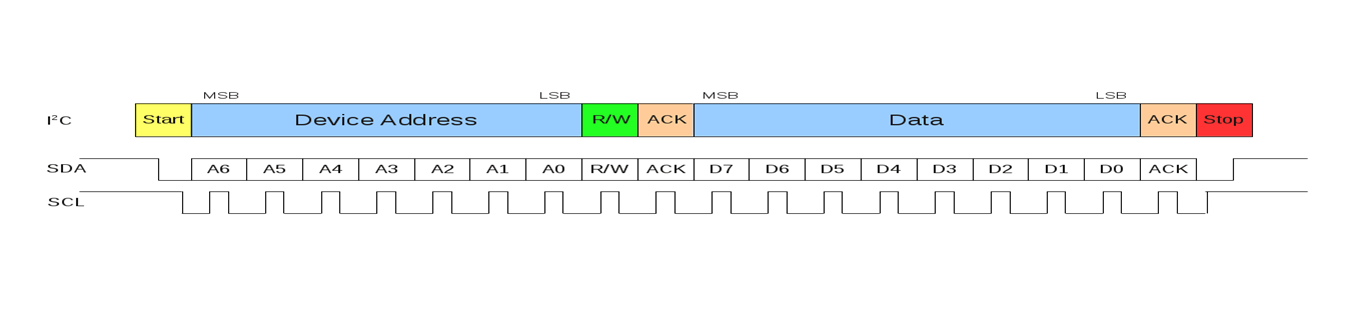


Figure 18 I2C

In general microcontrollers can be masters or slaves, but the computers can only be masters. Accelerometer and Gyroscope

See below an example of wiring and programming for a digital sensor.

Using this sensor, you can determine acceleration on all three axes and also the rotation angle. An important note is that it also measures gravitational acceleration so don`t be surprised to get values different from 0 when the sensor is not moving.

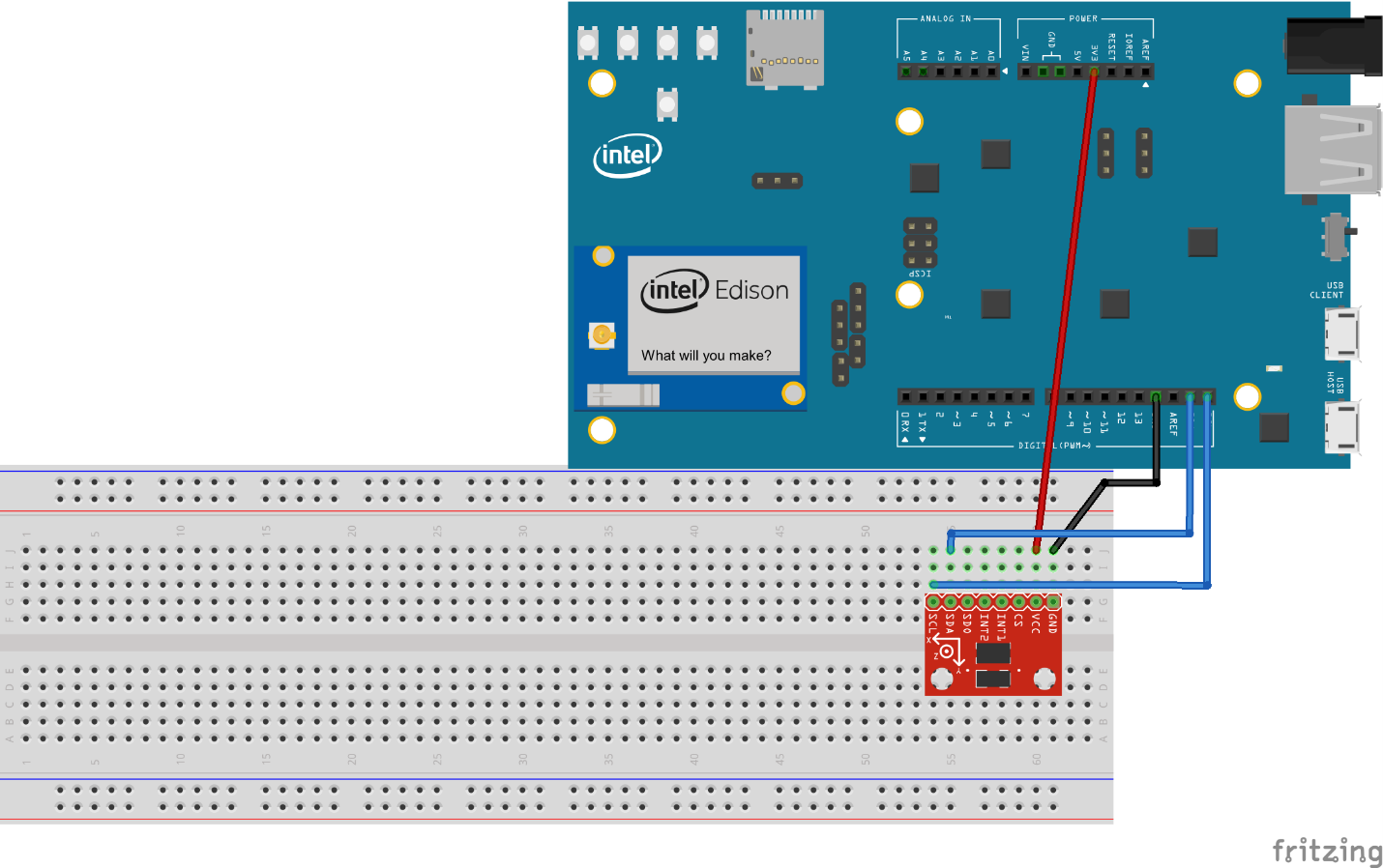


Figure 19 connection

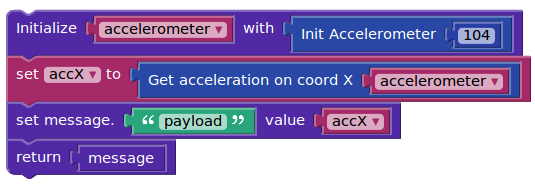


Figure 20 I2C in programing

**Peripherals**

## LED

An LED is a light emitting diode. A diode allows the current to pass only in one direction. Also, it has no resistance, which means it will be in a short circuit once it is wired to a circuit with no resistor. You will see when you look at the LED that it has two legs. One is longer, that one is usually the anode. This one has to be connected to the GPIO pin of the Edison. The shorter leg should to be connected to the resistor and then to the ground pin of the board.

Although the position of the resistor is not fixed, it can either connect the ground to the cathode or the anode to the GPIO pin, the cathode should be connected to the ground to obtain the usually desired behavior. That means that we want the LED to light when the GPIO is set to HIGH and not to light when the GPIO is set to LOW. If you put the legs the other way around, the effect will be the opposite.

## 7 segment display

It is an electronic component consisting of 7 LEDs. These have either a common cathode or common anode. In the first case, the LEDs need the value 1 on the pins to be alight, but in the second case, the VCC will be common. The latter situation means that the LEDs are alight when the value on the pins is 0.

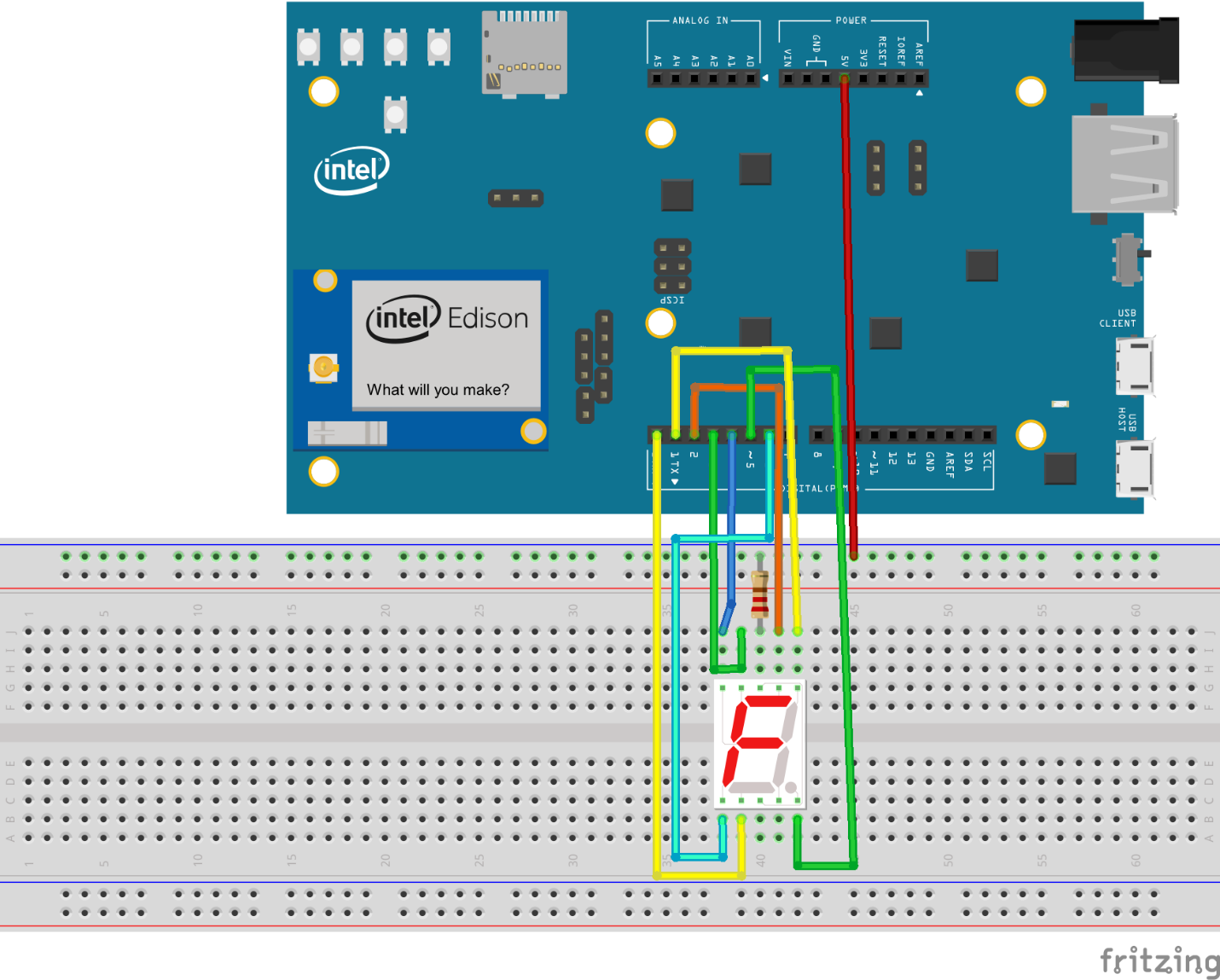


Figure 21 7 segment display connection

How do you connect it? In visual programming, there are special blocks for the piece. All you have to do is to look up your model's Datasheet AND insert the number of the pins for each segment.

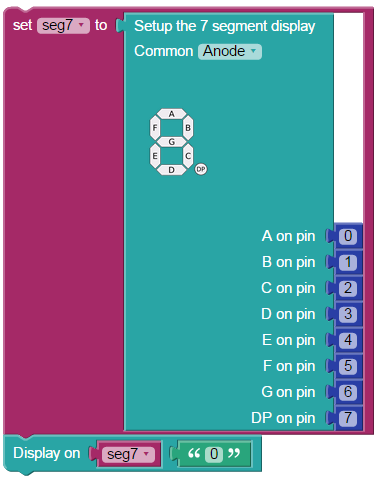


Figure 22 7 segment display in programing

## Shift register

It is a serial to parallel register. Being a register, it is actually a memory. The data stored inside goes from QA to QH. These are the GPIO pins. OE stands for output enable, it is used to disconnect the pins all at once. SER is the serials input pin, RCLK latch clock, SCLK is the serial clock and SRCLR is the clear pin. The data is easy to read, paralleled, but written by shifting it inside. When the serial clock switches from 0 to 1 the register reads the value in the SER, puts the value from QH into QH, QG into QH and so on until it's QAs turn to go into QB and SER is now stored into QA. You would need 8 cycles to change all the values from a register. The bits can be read all at a time, but not written this way.

Two shift registers can be connected together.

A shift register offers 8 outputs using only 2 lines: the clock and the SER. When SRCLR is 1, everything is cleared out. The shift register changes the values in the output as others come in, so this can be visible. To avoid this problem, good quality shift registers have a parallel to parallel register. This means that only when a cycle is complete the RCLK will switch and the values are given as output. This is why the latch clock is needed.

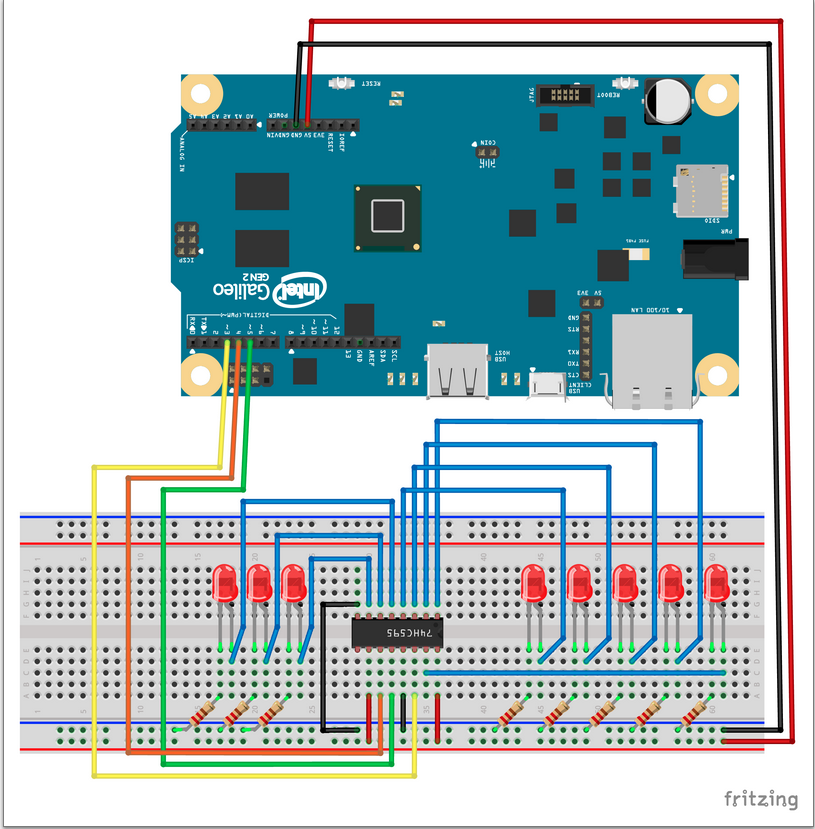


Figure 23 Shift register connection

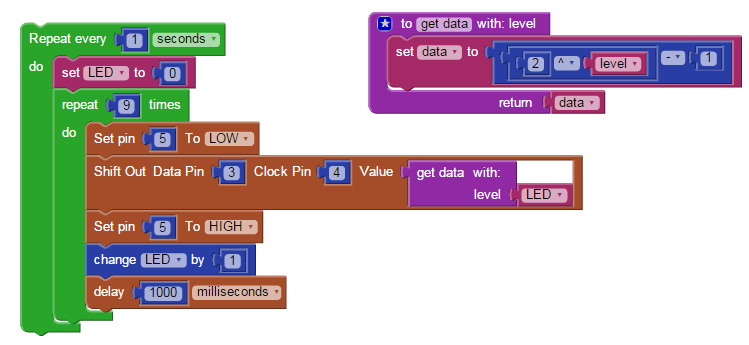


Figure 24 Shift register in programing

## LCD

An LCD with 16 colons and 2 rows needs 16 pins to be controlled. It can use either a 4 pin protocol or an 8 pin protocol. The former will need 12 pins and the last, all the 16 pins. There are 4 power pins, meaning two groups of GND and VCC, one group being used by the backlights. Any LCD uses a potentiometer to set the contrast. The enable pin of an LCD can be connected to the ground or to a microcontroller. There are also I2C LCDs that work with only 4 pins, have an integrated potentiometer and can be connected in the same circuit only if they have different addresses.

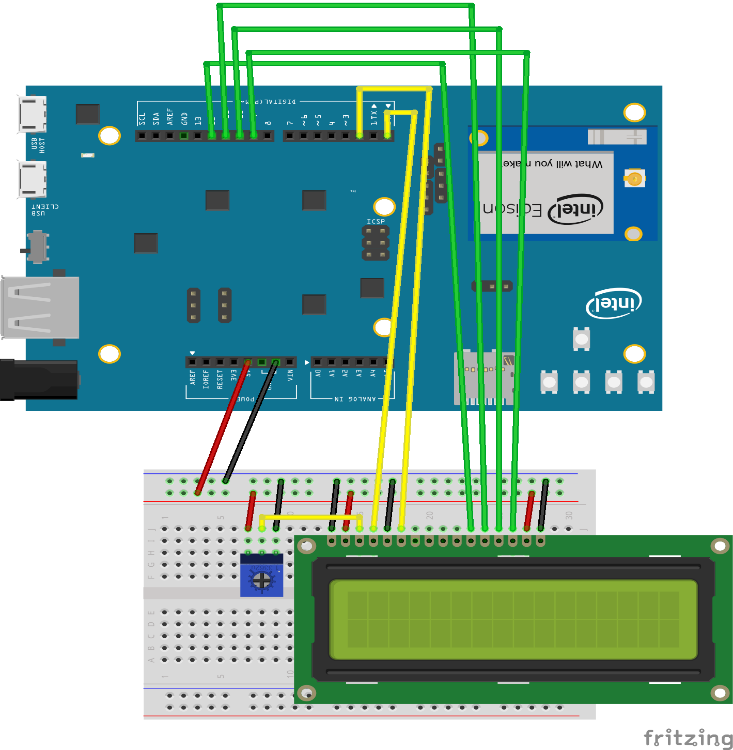


Figure 25 LCD connection

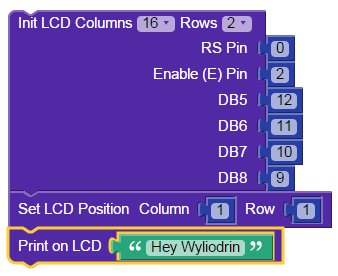


Figure 26 LCD in programing

This course explains what a web server is, how it works, protocols used in web programming and gives examples on building and implementing a web server, of creating dynamic web pages. It also introduces the students to AngularJS and JQuery.

## What's a web server and HTTP?

Any computer that can implement http or https is able to play the role of a web server. Http is a protocol, a way of communication which supplies web pages. It is pretty widely used and easy to implement. Through http you can transfer html and create simple user interfaces, it can implement Java Script and make more complicated web pages and it is available in most of the browsers. One of the great qualities of this protocol is that it replaced complicated and heavy displays with user friendly web pages.

How does it work? The browser sends a request to the server who searches the demanded page and returns it to the browser for the user. The request will consist of information about the kind of browser that is used, about the computer or about the document requested. It will have a method, a URL, a query string and the upload body in case you want data to be sent to the server.

The response will include the status, which tells the browser if the page was found or not (the errors among the 400s are about a not found page, 300 are redirections and 200s are confirmations of the page being found).

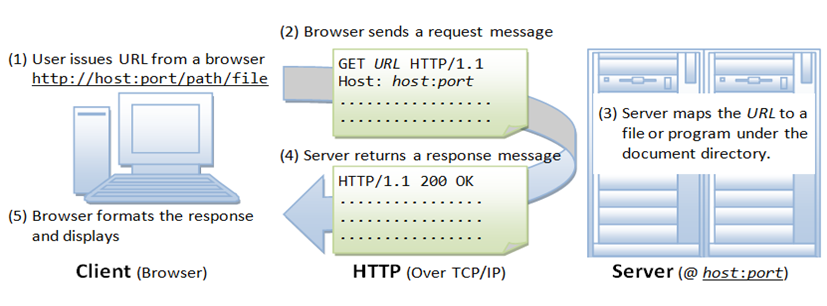


Figure 27 HTTP

## Https has two important security roles.

* It encrypts the data. The request and the response will be both encrypted on sending and decrypted when read.
* The server is always asked for a certificate of authenticity before it is asked for a page. This prevents against stolen data through false web pages.

What does a query consist of? It will always look like this: http://address:[port]URL?querystring. The port can be absent, in which case it will be 80 for http and 443 for https. It has to be specified if it is not one the two. Concerning the URL, when it is not written, the default will be /. The available methods in http are: get, post, put and delete. The main ones being the first two.

* Get method needs no upload body. It will only ask for data from the server and send only the headers, the address, the URL.
* Post sends important data to the server, which will be uploaded. Post has the role of modifying data on the server. The response of both these methods is the page and any additional information that was requested.
* Put is similar to post, only that in the semantic way, this method only creates an object on the server.
* Delete also plays a semantic part. It needs no upload body and it deletes objects on the server. The same action can be performed however using get.

On one server there can be more than one websites, which means that, if the host is not specified in the request, the response may not be the one the browser expects.

Also, the response may have more than text. Any additional feature: images, JavaScript objects and so on will need a new request, so the process will be slowed down.

## Webservers on gadgets using Wyliodrin

The boards are non-powerful computers. With wyliodrin there is no need to install any software or make any configuration on the boards to run a webserver on them.

To create a webserver in wyliodrin you will need a web node. The simplest way to use a web page in this particular way is to send static files. In the project files, create a new folder static. Everything inside it will be sent back to the browser by the server, regardless of the fact that they are html, Java Script or CSS files. Images can be added as well, but they will definitely make the process slower. There are other ways of adding an image. For example, by using a storage system and including the images from there. This method will solve speed and memory issues.

The web node: The route option is actually the URL. The webserver will be active when it stumbles upon the specified route. Afterword’s you choose the method, and write the port to setup the server. This port will only be used once, in the beginning.

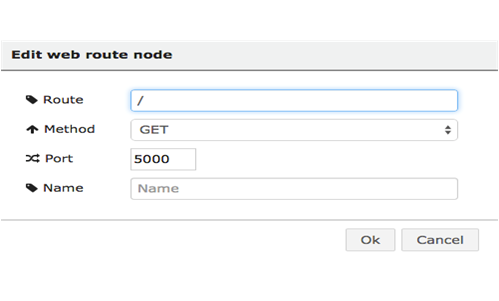


Figure 28 Webservers using Wyliodrin

The payload goes either in the query string for the get method or in the upload data for post. The message is built on this payload, on two mandatory variables: res which stands for the response and req. which is the request. Without the last two, the server won't be able to provide a response.



Figure 29 Webservers response

The web response node: The message received by this node comes from one web node. For a web response the simple way is to make a redirection. Which means, in the redirect field, you can write the path to one of the static files and the browser will be sent to this page. On top of these, you will need the board's IP address which might not be public unless it is in the same network with the web server.

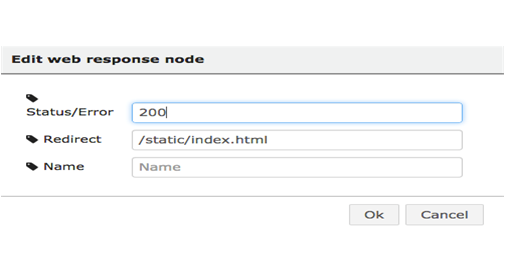


Figure 30 Webservers response using Wyliodrin

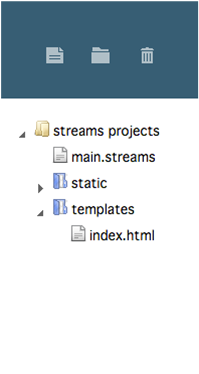


Figure 31 Webservers response files

As a solution, IOT servers have a public address. The port for these servers can be either 80 for http or 443 for https. The user accesses the public page, through the IOT server which is connected to Wyliodrin as well as the board. Now the problem with the board and the web being in the same network is solved as both can communicate with Wyliodrin.

Web templates: Just as for the static files, you will need a templates folder. This time, when you use the node, you don't need the whole path. You can only write the name of the file in the templates folder. What does the node do? It processes the response, meaning it loads the values plus the payload in it and sends it back to the browser. The values need to be in between two sets of curly brackets {{}}. Note that the values won't update unless the page is reloaded.

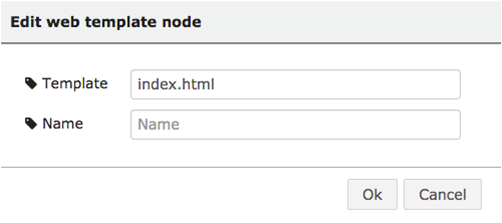


Figure 32 edit web template node

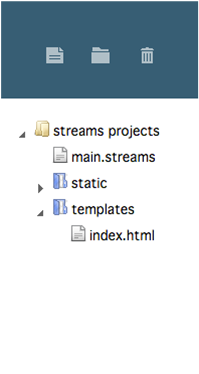


Figure 33 edit web template node files

## Web services

A long time ago, the web services were more complicated. Now the application only requests the web server for the data, and it is the browser's job to rearrange it so that it is in the right format for the application.

How to implement it into a Wyliodrin application? Using a simple web response and web server node, you send a static page to the user and each time you make a query, instead of a template, you use a web response node and send the payload to the browser, which can be a number, an object or anything else.

## JQuery

There is a library called JQuery, based on JavaScript, thus available in any browser, which can make function calls to the server.

Case study: You have the following situation: you change the payload into a variable which stores values from a sensor. You want this variable to be shown in your web page. Practically, when an API gets called, what you will do, will be to make a get request to the server using the web address that you want with the URL /sensor. The web page will send values that you will store in a variable in your html file.

## Web sockets

A web socket is based on the http or https protocol. It builds a connection between the browser and the server, so that either one can send data. When the browser makes a request, the server recognizes the socket and doesn't close the connection. The two parties send the packages they need to send. If the server does not know how to work with sockets, the socket io will go back to querying.

## AngularJS

AngularJS is a library through which you can build browser applications. In the next example, every web node will create a new socket and serve a static web page. If you include in the response a variable, and this variable changes, wyliodrin controller will be notified every time this kind of novelty appears and AngularJS will replace the old value of the variable with the new one, creating a dynamic web page.

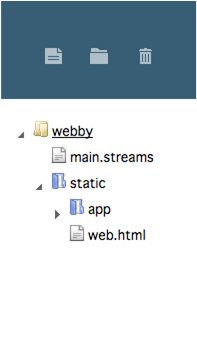


Figure 34 AngularJS files



Figure 35 AngularJS files and web response



Figure 36 AngularJS in code

# CHAPTER 2: Used Martials and its Features

In our project we have used some specific martials … in this chapter we will discuss these martials and its feature. Our project consists of main components including RaspberryPi, H-bridge and Wild Thumper.

## 2.1 RaspberryPi

The Raspberry Pi is a single-board computer with wireless LAN and Bluetooth connectivity developed by the Raspberry Pi Foundation in the UK to promote basic computer science teaching in schools and developing countries. The original model was much more popular than anticipated, selling uses such as robotics outside of its target market. This is now commonly used also in research projects, because of its low cost and portability, for example for weather monitoring. It does not contain peripherals (such as mice and keyboards), or cases. However, some official and unofficial packages included several accessories. The Raspberry Pi used in our project is the Raspberry Pi 3 which is the third and most recent Raspberry Pi generation. Just since February 2016, it replaced the Raspberry Pi 2 Model B.

|  |  |
| --- | --- |
| https://ram-e-shop.com/wp-content/uploads/2018/09/raspberry_pi_3b.jpg | https://ram-e-shop.com/wp-content/uploads/2018/09/raspberry_pi_new.jpg |
| Figure 37 Raspberry Pi 3 | |

## Features:

* Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
* 1GB RAM
* BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
* 100 Base Ethernet
* 40-pin extended GPIO
* 4 USB 2 ports
* 4 Pole stereo output and composite video port
* Full size HDMI
* CSI camera port for connecting a Raspberry Pi camera
* DSI display port for connecting a Raspberry Pi touchscreen display
* Micro SD port for loading your operating system and storing data
* Upgraded switched Micro USB power source up to 2.5A

## GPIO

An important feature of the Raspberry Pi is the row of GPIO (general input / output) pins along the board's top edge. Both existing Raspberry Pi boards (unpopulated on Pi Zero and Pi Zero W) have a 40-pin GPIO header. Until the Pi 1 Model B+ (2014) the boards had a shorter header of 26-pin.

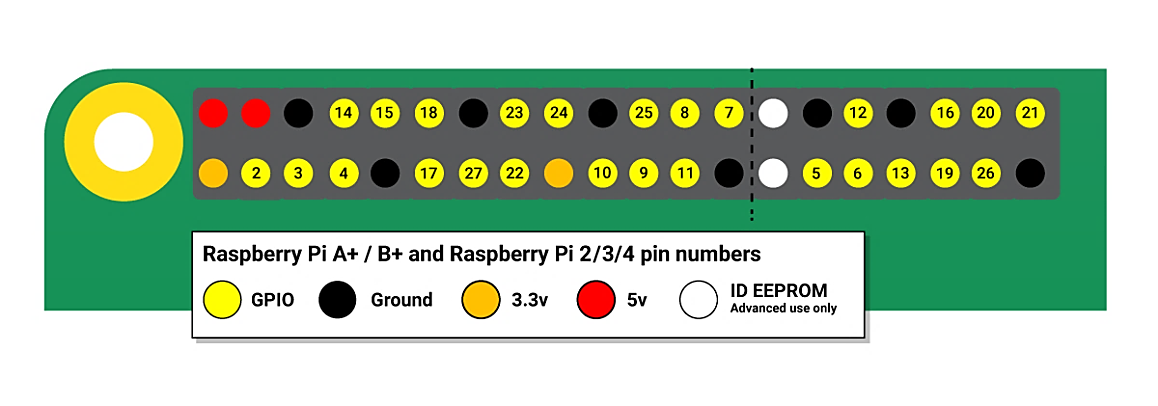


Figure 38 Raspberry Pi Pin Numbers

Any of the GPIO pins can be designated as an input or output pin (in software) and used for a broad range of purposes. Note that GPIO pins are not numbered in numerical order; GPIO pins 0 and 1 are on the board (physical pins 27 and 28), but are reserved for advanced use.

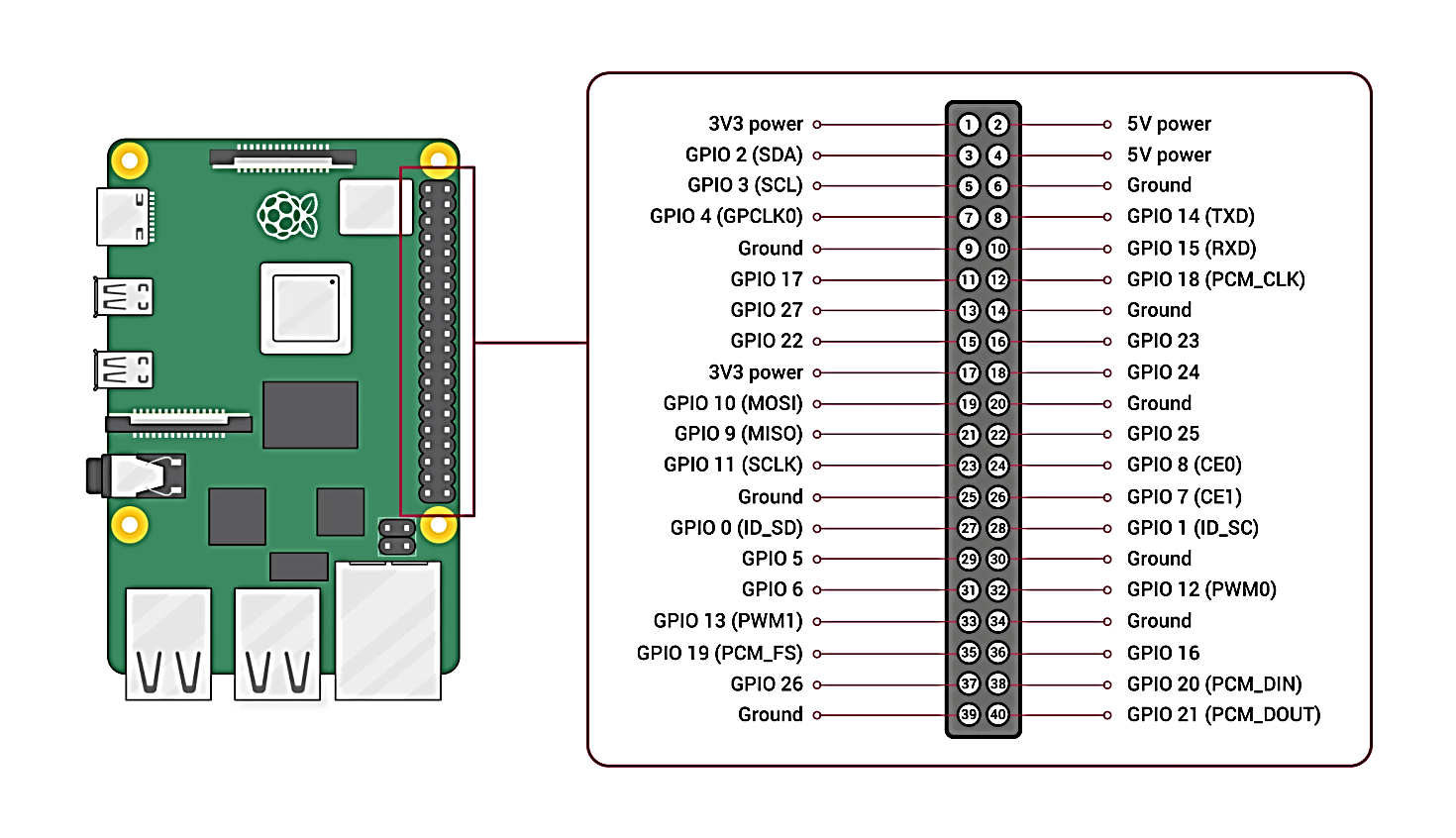


Figure 39 Raspberry Pi Pin Assignment

## Voltages

On the board are two 5V pins and two 3V3 pins, as well as several ground pins (0V), which are unconfigurable. The remaining pins are all 3V3 pins with general purpose, meaning outputs are set to 3V3, and inputs are sensitive to 3V3.

## Outputs

They can be set to high (3V3) or low (0V) with a GPIO pin designated as an output pin.

## Inputs

## A high (3V3) or low (0V) GPIO pin designated as an input pin may be read. This is made easier using internal pull-up resistors or pull-down resistors. GPIO2 and GPIO3 pins have fixed pull-up resistors but this can be configured in software for other pins.

## More about GPIO

The GPIO pins can be used with a number of alternative functions as well as simple input and output devices; some are available on all pins, others on different pins

* PWM (pulse-width modulation)
  + Software PWM available on all pins
  + Hardware PWM available on GPIO12, GPIO13, GPIO18, GPIO19
* SPI
  + SPI0: MOSI (GPIO10)
  + MISO (GPIO9)
  + SCLK (GPIO11)
  + SPI1: MOSI (GPIO20)
  + MISO (GPIO19)
  + SCLK (GPIO21)
* I2C
  + Data: (GPIO2)
  + Clock (GPIO3)
  + EEPROM Data: (GPIO0)
  + EEPROM Clock (GPIO1)
* Serial
  + TX (GPIO14)
  + RX (GPIO15)

## GPIO pinout

It's important that you know which pin is which. Some people use pin labels (such as the RasPiO Portsplus PCB, or the Raspberry Leaf printable). On the Raspberry Pi a handy reference can be accessed by opening a terminal window and running the pinout command. The GPIO Zero Python library provides this tool, which is installed by default on the Raspberry Pi OS desktop image but not on Raspberry Pi OS Lite.

## 2.2 H-bridge

An H-bridge is an electronic circuit that switches the polarity of a voltage applied to a load. These circuits are often used in robotics and other applications to allow DC motors to run forwards or backwards.

The H-bridge used in our project is (BTS7960B H-bridge 43A) high-power motor driver Module (For Single Motor). This driver uses Infineon chips BTS7960 composed of high-power drive full H-bridge driver module with thermal over-current protection. Double BTS7960 H-bridge driver circuit, with a strong drive and braking, effectively isolating the microcontroller and motor driver! High-current 43A

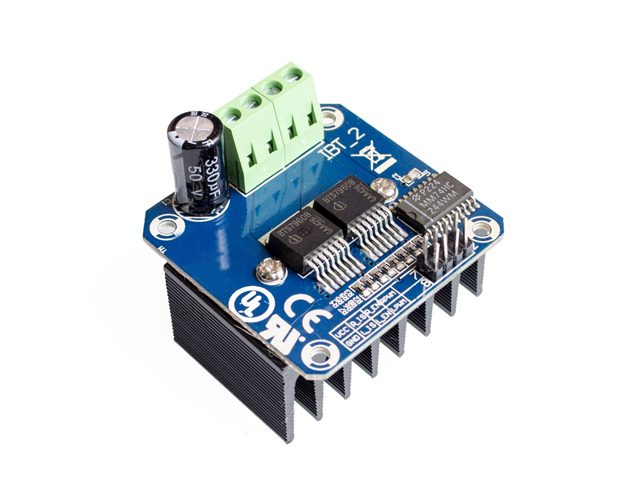


Figure 40 BTS7960B H-bridge

## Features:

* Double BTS7960 large current (43 A) H bridge driver.
* 5V isolate with MCU, and effectively protect MCU.
* 5V power indicator on board.
* Input supply voltage 5.5V to 27V.
* Voltage indication of motor driver output end.
* Just need four lines from MCU to driver module (GND. 5V. PWM1. PWM2).
* Able to reverse the motor forward, two PWM input frequency up to 25 kHz.
* Two heat flow passing through an error signal output.
* Isolated chip 5V power supply (can be shared with the MCU 5V), can also use the on-board 5V supply.

## 2.3 Wild Thumper

The robust “Wild Thumper” 6WD All Terrain Chassis for robotic applications is made from 2mm thick anodized aluminum plates with stainless steel and nickel plated brass fittings. Custom electronic boards and sensors can be mounted easily on the many 4mm holes which are arranged in a 10mm grid. The Wild Thumper is equipped with 6 powerful steel gearbox motors (1:35 ratio, 4kg/cm stall torque per Motor), spiked tractor tires and a “Super Twist” suspension system to keep all wheels on the ground. This chassis will let your robot go almost anywhere in rough terrain and drive over obstacles. The platform is perfect for robotic developers and student projects. Between the wheels are two large Battery compartments which can hold up to 4 x 7, 2 V RC Car subs C Battery Packs.



Figure 41 Wild Thumper

## Specifications:

• Dimensions: (L x W x H) 430 x 310 x 135 mm  
• 2mm thick anodized aluminum plates  
• Adjustable “Supertwist” suspension system  
• 6 spiked tractor tires.   
• Rotation speed: 295 rpm

# CHAPTER 3: Code Explanation

In this chapter, we will explain the implemented code by which the project is done. In this project, we classified the coding files into two section … the first one is the one that cares about the HTTP server and how it interact with the user and with the motor, the second section that cares about the programming of the RaspberryPi and the motor.

## 3.1 HTTP Server

In this section talking about the programming code by “JAVA” Programming Language of the HTTP Server. This Section contains one file which is the named file called “HTTP\_Server”. But Before Explain the code … we must know the importance of the server in our project. It’s simply to send to the RaspberryPi (C++) Code a request with direction of the movement the user wants to move the body. If the user gives an order, the server replies with that order. It makes that cycle 10 times per one Second. It means that the user has the chance to give an order ten times in every second by any movement as Left, Right, Forward, Backward or stop from movement; so the (C++) code send a request for the server to give it any order and the reply of the java code includes the order of the movement.

At the First of the java code … we start to import some java files that we will use indeed in the code.

|  |  |
| --- | --- |
|  |  |
| Figure 42 Java code: import section | |

At First we declared two variables that we will be used later in the java code … those variables are a Text field named text1 and the other one is a Panel named contentPane. After that we create a new object from HTTP\_Server class called frame and make it as visible. In this object we just call HTTP\_Server Function that will be explained later in this code.

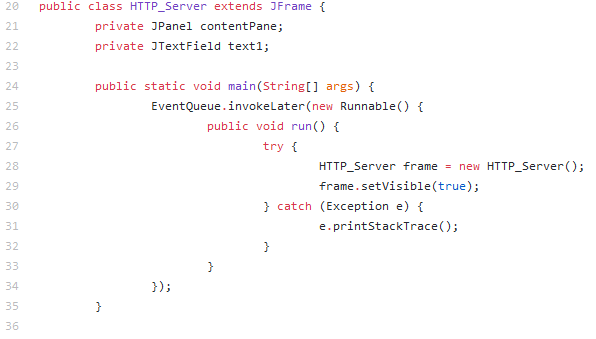


Figure 43 Java code: Create new HTTP\_Server

Then, we have defined HTTP\_Server function. This function makes a frame that Exit on Close and set some properties for it including the bounds. After that, make a new panel named content Pane with setting some properties to it including border and layout. Later, make a variable named server from HTTP-Server Type.

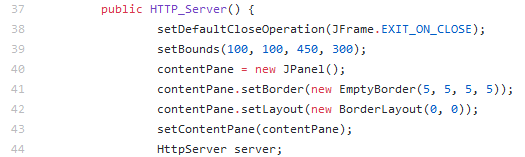


Figure 44 Java code: HTTP\_Server Function

Simply, Here If everything is gone very well … Then we just create a default executor and start the server. On the other hand, if there is any error expected happened … Then Show Error Message.

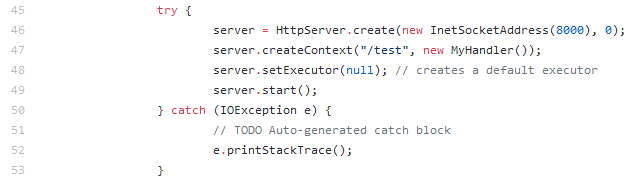


Figure 45 Java code: Server Start

Later, We make textbox called text1 and set in it a text by “STOP” while customization of some of its properties like add it at the center of the content Pane and make it focus.

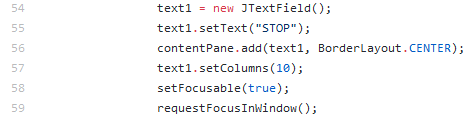


Figure 46 Java code: Textbox Creation

In this code we make a Listener for the key entry on which if the user pressed or typed by any Button … it save the key. While if the key is released then don’t save it.



Figure 47 Java code: Key Listener

Here we make some condition for the order … if the user press or type on the UP keyboard button then set text for text1 textbox as “UP”, While if the user press or type on the DOWN keyboard button then set text for text1 textbox as “DOWN”, While if the user press or type on the LEFT keyboard button then set text for text1 textbox as “LEFT”, While if the user press or type on the RIGHT keyboard button then set text for text1 textbox as “RIGHT” Or if nothing of these happened … then set text for text1 textbox as “STOP”.

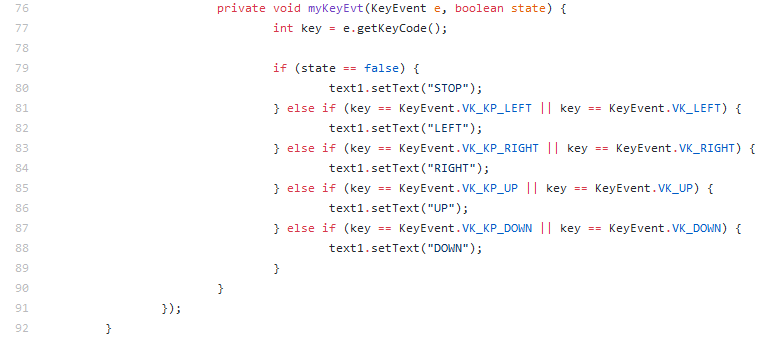


Figure 48 Java code: response status code

Finally, here when C++ code request an order … just get the order from the text1 text box and send the text as a reply for the code.

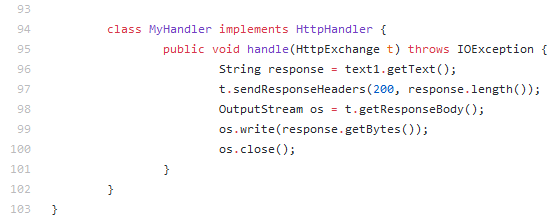


Figure 49 Java code: send reply

## 3.2 RaspberryPi

In this section talking about the programming code by “C++” Programming Language of the RaspberryPi module and the motor. This Section contains three files which are the Motor header file, the Motor Function description file and finally and absolutely the Main project file. But Before Explain the code … we must shout out that we used a method in code called Pulse-width modulation (PWM) so we must have an small brief on the PWM and its principles in order to understand the way of design is made and has been coded by.

## 3.2.1 Pulse-width modulation (PWM)

Pulse-width modulation (PWM) is a strategy to reduce an electrical signal's average power by essentially cutting it into discrete sections. The average voltage (and current) value fed to the load is managed by switching the switch among both supply and load on and off at a quick rate. Compared to the off periods the longer the switch is on, the higher the total power supplied to the fee. Along with maximum power point tracking (MPPT), it is one of the primary ways of reducing solar panel output to that which a battery can use. PWM is especially suitable for running inertial loads such as motors which are not as easily influenced by this discrete switching because they have inertia for slow reaction. The frequency of PWM switching must be high enough not to affect the load, which is to say the resulting waveform perceived by the load must be as smooth as possible.

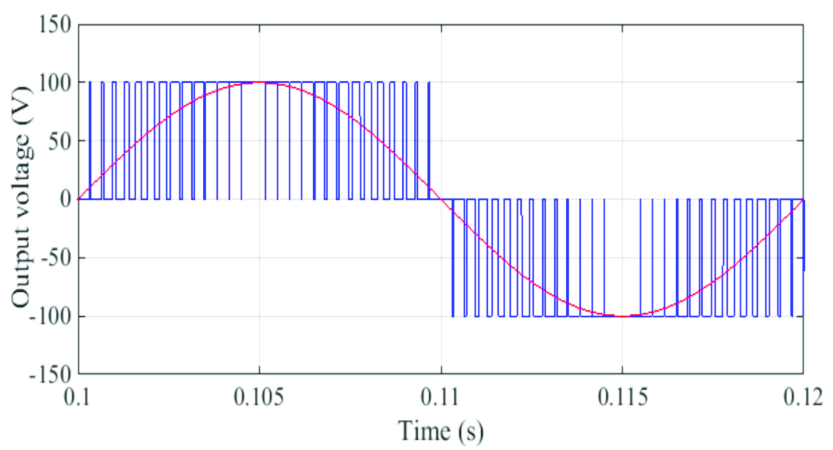


Figure 50 Ideal pulse-width modulation (PWM)

Depending on load and application, the rate (or frequency) at which the power supply has to switch can vary considerably. The main advantage of PWM is that it has very low power loss in the switching devices. There is virtually no current when a switch is off, and when it is on and power is transferred to the charge, there is almost no drop in voltage across the switch. Consequently, power loss, being the product of voltage and current, is close to zero in both cases. PWM also works well with digital controls which can easily set the required duty cycle because of their on / off nature.

## 3.2.2 Motor header code

This code file named “motor.h” and it is responsible for the declaration of the function used in the main code. In the first, we made class by ‘Motor’ name that we can take object from it anytime we needed to. Any Class consists of public and private sections … For the public one, we declare function for motion which are forward, backward, turn Right, turn Left and stop. All those function doesn’t take or return any attribute. At last, in the public section, we also declare a function called ‘motor’ that takes six attributes which are pwmRightPin1, pwmRightPin2 those are the number of the Pulse Width Modulation Pin for the right side of the motor, pwmLeftPin1, pwmLeftPin2 those are the number of the Pulse Width Modulation Pin for the left side of the motor, pwmMinVal, pwmMaxVal those are the number of the Pulse Width Modulation value of the motor … all those attributes are from integer data type.

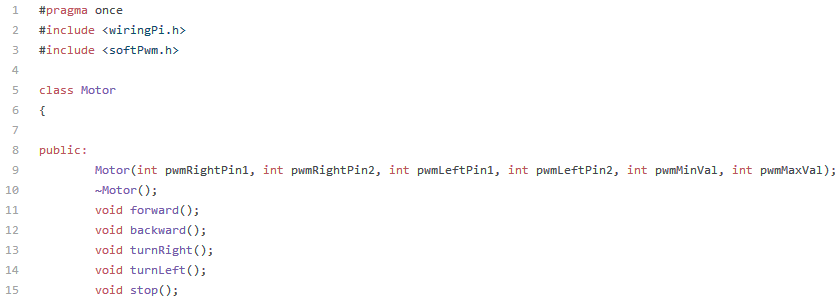


Figure 51 Motor header code: Public section

After Public section, it is the turn for the private one … we declare function for motion which are left Side Forward, left Side Backward, right Side Forward, right Side Backward, left Side Stop and right Side Stop. Also we have declared a function called ‘init’ for initialization of the motor. All those function doesn’t take or return any attribute as those in the public section. We also declare some variables for the Pulse Width Modulation which are pwmRightPin1, pwmRightPin2 … those are for the number of the Pulse Width Modulation Pin for the right side of the motor, pwmLeftPin1, pwmLeftPin2 … those are for the number of the Pulse Width Modulation Pin for the left side of the motor, pwmMinVal and pwmMaxVal … those are the number of the Pulse Width Modulation value of the motor. All those variables are from integer data type.

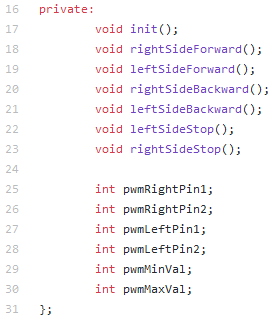


Figure 52 Motor header code: Private section

## 3.2.3 Definition of Motor Function Code

This code file named “motor.cpp” and it is responsible for the Definition of the function used in the main code and that was declared in the header file. In the first, we include the header file which is called ‘motor.h’ that has the declaration of the functions needed to be defined. We make an object from class called “motor” … in this object “Motor” Function we assigned the variables of the PWM pins and values which we have initialized before including pwmRightPin1, pwmRightPin2, pwmLeftPin1, pwmLeftPin2, pwmMinVal and pwmMaxVal. We assign these variables with the attributes taken by the function including pwmRightPin1, pwmRightPin2, pwmLeftPin1, pwmLeftPin2, pwmMinVal, pwmMaxVal assigning them in the variables in the same order. After assigning the pins and PWM values … we call the initialization function that is called ‘init’. That function we will explain later in this code file.

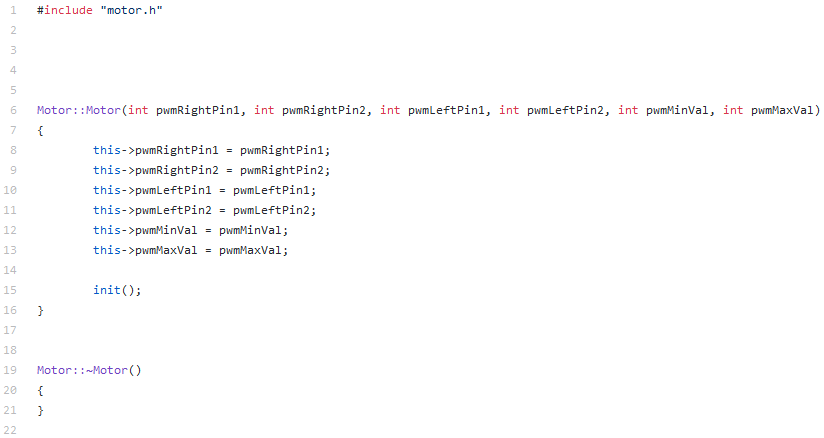


Figure 53 Definition Motor Code: motor function

After that, we define the motion function beginning with the forward function; to make the body go forward we make the right side and the left one go forward as shown in the figure (55) … so we call the functions right Side Forward and left Side Forward. Those functions will be explained later in this code file.

|  |  |
| --- | --- |
|  |  |
| Figure 54 Definition Motor Code: forward function | Figure 55 Forward Motion |

Then, we continue defining the motion function of backward function; to make the body go backward we make the right side and the left one go backward as shown in the figure (57) … so we call the functions right Side Backward and left Side Backward. Those functions will be explained later in this code file.

|  |  |
| --- | --- |
|  |  |
| Figure 56 Definition Motor Code: backward function | Figure 57 Backward Motion |

Then, we continue defining the motion function of turn Right function; to make the body turn right we make the right side go backward and the left one go forward as shown in the figure (59) … so we call the functions right Side Backward and left Side Forward. Those functions will be explained later in this code file.

|  |  |
| --- | --- |
|  |  |
| Figure 58 Definition Motor Code: turn Right function | Figure 59 Turning Right Motion |

Finally, we continue defining the motion function of turn Left function; to make the body turn left we make the right side go forward and the left one go backward as shown in the figure (61) … so we call the functions right Side Forward and left Side Backward. Those functions will be explained later in this code file.

|  |  |
| --- | --- |
|  |  |
| Figure 60 Definition Motor Code: turn Left function | Figure 61 Turning Left Motion |

After defining the motion function of turn Right function, we define the initialization function. We define the mode of the four PWM pins (Two Left and Two Right) as output pins. Also, we create the voltage range that will be deal with by assign the four pins in ranges of the minimum and maximum PWM values.

Note that this initialization is called only when we make an object of motor class; or we can say that the initialization is called one and only at every motor created. In this code … we have only one motor so this function is called only one time.

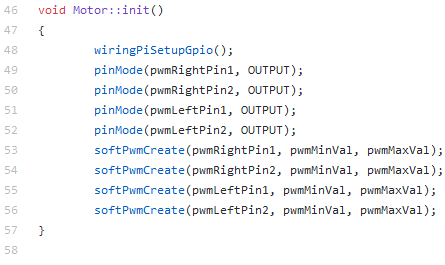


Figure 62 Definition Motor Code: init function

After defining the initialization function, we begin to define the movement of each side forward and backward. We define the movement of Right side forward by assigning high value on Right Pin no. 1 while assigning low value on Right Pin no. 2, Right side backward by assigning low value on Right Pin no. 1 while assigning high value on Right Pin no. 2, Left side forward by assigning high value on Left Pin no. 1 while assigning low value on Left Pin no. 2, and finally Left side Backward by assigning low value on Left Pin no. 1 while assigning high value on Left Pin no. 2.

|  |  |
| --- | --- |
|  |  |
| Figure 63 Definition Motor Code: Right side functions | Figure 64 Definition Motor Code: Left side functions |

After defining the movement of each side forward and backward, we define the stop of the movement of Right side, Left side and the whole body. First we begin with the stop movement of the left side by the same logic of movement the left side but by assigning low value on both Left pins number one and two at stopping left side.

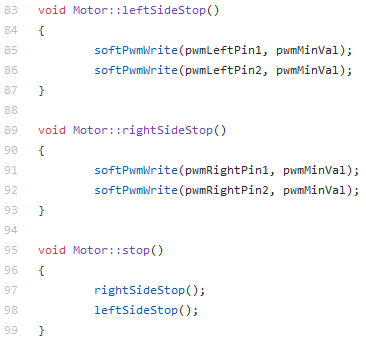


Figure 65 Definition Motor Code: stop functions

Then, we begin with the stop movement of the right side by the same logic of movement the right side but by assigning low value on both right pins number one and two at stopping right side.

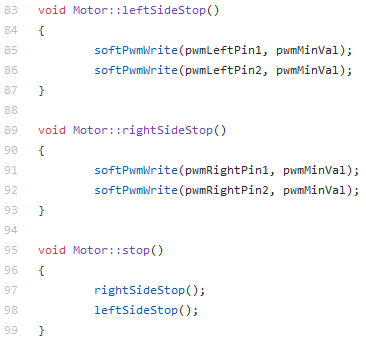


Figure 66 Definition Motor Code: stop functions

Finally, we begin with the stop movement of the whole body by stopping the left and right side from movement by calling the right Side Stop and left Side Stop Functions that where explained before in this code file.

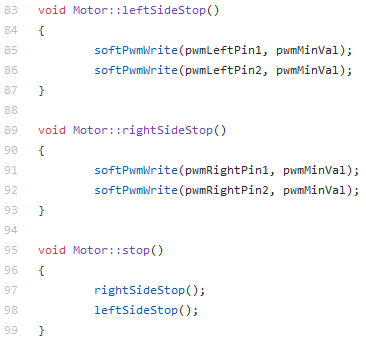


Figure 67 Definition Motor Code: stop functions

## 3.2.4 The Main Project Code

This code file named “pro1.cpp” and it is responsible for the main program. In the first, we include the header file which is called ‘motor.h’ that has the declaration of the functions needed to be defined and some other include files we in need for them. In the main block, we create an object from “motor” Class with assigning in it the pin numbers and values required in motor function. We assigned the pins number (20, 21, 13 and 19) as motor pins, assigning ‘0’ for minimum value of PWM value and ‘100’ for maximum value of PWM value. We also declared some variable that will be used including variable named response for the HTTP response, variable named body response for the receiving response status code and, variable named body response for the receiving response status code.

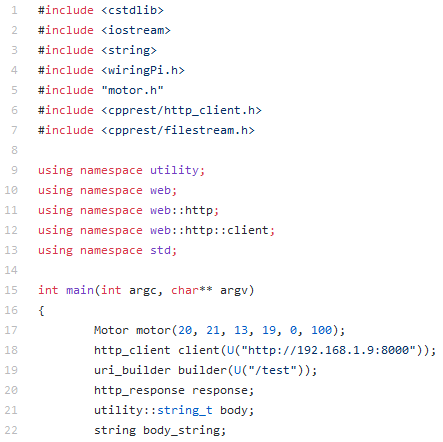


Figure 68 Main Project code: initialization

In an infinite loop – The program repeated infinite number – we get the Received response status code and assign the value a variable called body that was defined before. If this process doesn’t go well – Any error Occurred – then stop the motor immediately … if the process of getting the code goes good – No error occurred – then do the following code.

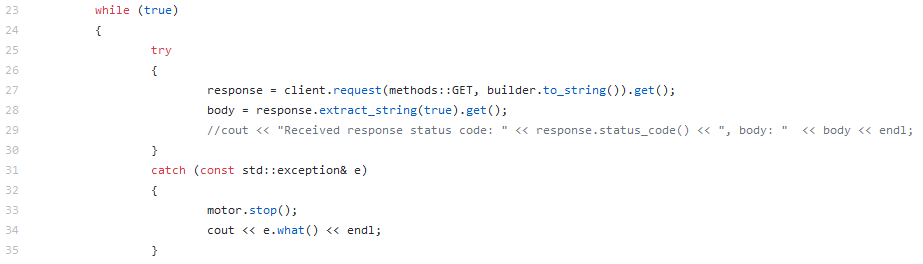


Figure 69 Main Project code: Get response status code

Here, we assign what we get of response status code and check whether the response status code is equal UP then the body move forward, or it is equal DOWN then the body move backward, or it is equal Right then the body turn right, or it is equal LEFT then the body turn left … OR nothing of those condition then the body will stop.

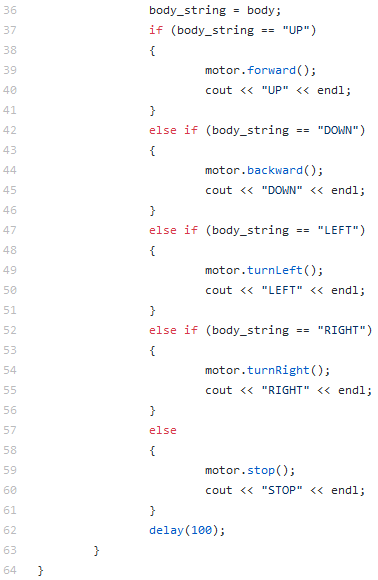


Figure (70) – Main Project code: Condition block

Figure 70 Main Project code: Condition block

# CHAPTER 4: Simulation

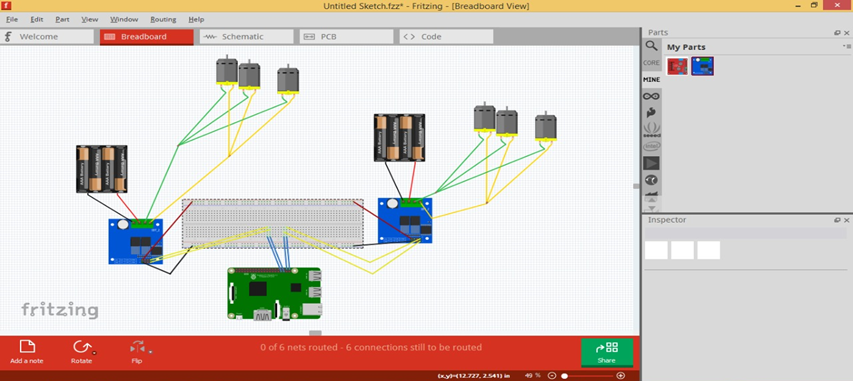
****

Figure 71 Simulation

# CHAPTER 5: Conclusions and Future Work

## Conclusions

We achieved our target by leaning the main concepts of IOT that helped us in our project and it mentioned in the introduction after that we explained the steps that we made it from explaining the hardware parts that we used and their job in the project and their details like dimensions , speed of rotation and the sensors that we used in project too and how it works and explaining codes that we used and their effect on hardware body like motion and making the simulation to test it the output of the project and make sure that the output will be acceptable or not and after all our project is one of the examples that help the others to learn the concept of IOT and how it works and the important of the IOT in our life to make it easier.

## Future Work

This project aims to make a modest security IOT device. Thus, live capturing is indeed necessary to achieve that goal. A camera shall be added to the robot car, so it provides this live capturing feature and make this robot more reliable.

# CHAPTER 6: References and Appendices

1. **https://ram-e-shop.com/product/raspberry-pi-3/**
2. **https://ram-e-shop.com/product/kit-driver-dc-bts7960/**
3. **https://ram-e-shop.com/product/ro-base-rs003s34/**
4. **https://github.com/AhmadSakr/CloudRobot/blob/master/README.md**
5. **https://www.raspberrypi.org/documentation/usage/gpio/**