SRH University Heidelberg

# Master's Thesis

Robot Development


| | |
|---|---|
| Author: | Emran, Fahad Bin |
| Matriculation Number: | 11010458 |
| Address: | Ringstraße 37<br>69115 Heidelberg<br>Germany |
| Email Address: | ayonfahad@gmail.com |
| Supervisor: | Prof. Dr. Achim Gottscheber |
| Begin: | 09. January 2019 |
| End: | 10. July 2019 |

# Acknowledgement

Dear reader,

This is work for my master thesis and conclusive work for the Master of Engineering in Information Technology at SRH Hochschule Heidelberg.

For the successful outcome of this thesis, it required a lot of support, patience, and guidance from many inspiring individuals. First and foremost, I would like to extend my sincere gratitude to my thesis supervisor Prof. Dr. Achim Gottscheber of the School of Engineering and Architecture at SRH Hochschule for his efforts, guidance, and support throughout the thesis period.

I wish to express my special thanks to Mr. Klaus Friedrich, for his expertise suggestion and help regarding printing the parts in the 3D printer at the mechanical lab. I would also like to thank Mr. Carlo Vondano and Mr. Pablo Marcel Disson for their expert opinion regarding choosing the parts and components.

Finally, I wish to offer my gratitude to my parents, family, and friends for their continuous support throughout my studies. The accomplishment so far would not have been possible without their belief in me. This success is dedicated to all these motivating individuals who were part of this journey.

Kind regards,

Fahad Bin Emran

# Declaration of Authorship

I at this moment declare that the thesis submitted is my unaided work. All direct or indirect sources used are acknowledged as references.

I am aware that the thesis in digital form can be examined for the use of unauthorized aid and to determine whether the thesis as a whole or parts incorporated in it may be deemed as plagiarism. For the comparison of our work with existing sources, I agree that it shall be entered in a database where it shall also remain after examination, to enable comparison with future thesis submitted. Further rights of reproduction and usage, however, are not granted here. This paper was not previously presented to another examination board and has not been published.

| _____ | _____ |
| --- | --- |
| First and last name | City, date and signature |

# Abstract

This thesis paper follows developing an autonomous robot that follows the criteria of Eurobot2019 and performs the first mission that is classifying the atoms by using the DC, Servo motors, batteries, Arduino Uno microcontroller, Raspberry Pi SoC computer and Pi camera mounted on the robot chassis. In terms of detecting the object from the Camera, the thesis follows the use of TensorFlow neural networks and covers from training the object detection classifier in Windows10 to run the object detection classifier model in Raspberry Pi.

# Kurzfassung

Diese Arbeit folgt der Entwicklung eines autonomen Roboters die folgenden Kriterien von Eurobot2019 und führt die erste Mission, die die Klassifizierung der Atome durch die Verwendung der DC, Servomotoren, Batterien, Arduino Uno Mikrocontroller, Raspberry Pi SoC Computer und Pi Kamera auf dem Roboter-Chassis montiert. Im Hinblick auf die Erkennung des Objekts von der Kamera, die These folgt die Verwendung von TensorFlow neuronale Netze und deckt von der Ausbildung der Objekterkennung Klassifikator in Windows10, um die Objekterkennung Klassifikator Modell in Raspberry Pi.

# Contents

# Chapter 1
# Introduction

## 1.1. Introduction

We are living in a fast-paced world where technology advancement is going forward at a rapid rate. This rate is higher in robot development than most of the other cases. A robot is a machine capable of doing complex actions or follow commands.

During the industrial revolution era, humans began to research and find out a way to come up with various types of machinery that can perform the labour intensive task more effectively and efficiently than human workers. They came up with building the industrial robots that mitigate the operational hazards and boosted the output production. From there we have improved a lot in robot development. Apart from medical robots, we are sending the robots to remote places where the human can not explore and get first-hand information and intervention.

Nowadays, we are more focusing on mobile robots as we are trying to make our lives as easier as possible. Mobile robots are miniature robots capable of doing specified tasks while having fewer resources. The autonomous property adds an extra feature to the of the robot providing a higher level of decision making without any external influence. The recent advancement of IoT has opened the door for further development in home autonomous robots. For example, using the IoT feature in an autonomous mobile robot one can manage and do particular tasks without the user's presence near the action area.

Building an autonomous robot was expensive in previous days because of the information and knowledge were classified to the public. Hardware and the types of equipment were not budget friendly either. However, in recent years the price of the materials and hardware for robot development has decreased significantly and now one can think of building a mobile autonomous robot for home furthermore commercial use.

## 1.2. Motivation and Goal

The motivation for the thesis is to develop an autonomous robot and the goal is to perform a specified task of Eurobot2019. The robot development field was kind new to me. In October of 2018, I have performed a group project where we have built two autonomous robots performing three tasks of Eurobot2018. From there, I have learned the basics of Robot development which made me very curious about the topic. While choosing my master thesis topic, I wanted to further improve my knowledge and skills in this particular field. In robot development project

work, we were using Arduino microcontrollers, sensors and other mechanical hardware. In this master thesis, I wanted to introduce the Raspberry Pi SoC computer and instead of IR sensors, ultrasonic sensors, RGB sensors use the PiCamera which have enabled me to do object detection using a machine learning framework. In short, the goal is mentioned below

- Components selection for the task.
- Construct and optimize the electronic and mechanical pieces of equipment.
- Walkthrough the implementation steps and algorithms followed in the development.
- Troubleshooting problems faced in the development phase.
- Learn about TensorFlow neural network implementation.

## 1.3. Thesis outline

**Background** describes the hardware and software that are going to be used in the thesis.

**Eurobot2019** describes introduction, rules and missions.

**Implementation** describes the implementation steps that were involved in this thesis.

**Results** describe the result we are getting from the robot we have developed.

**Discussion and conclusion** describe the general discussion, notable problems, conclusion and future works.

# Chapter 2
# Background

## 2.1. Raspberry Pi

Raspberry Pi is a card-sized SoC capable of doing small to medium tasks a traditional desktop computer can perform. Funded by Raspberry Pi foundation, the goal of the charity foundation is to contribute to the advancement in the field of computer science among young and adults. That is is one of the reasons that one can get a Raspberry Pi computer by spending around $35 and start developing some projects most commonly in IoT applications. There is a model named Pi Zero that costs around $5 with reduced facilities.

This third bestselling computer brand in the world follows ARM CPU architecture and also has onboard GPU. The ARM architecture follows on RISC architecture. Unlike CISC architecture, this ISA has less CPI. A typical Raspberry Pi computer has no peripherals connected. It has a 900 MHz 32bit quadcore ARM processor, onboard HDMI port to connect it to a display device, 3.5 mm audio jack port, USB port, Ethernet port, 40 GPIO pins (pin layout is shown in Figure 2-2), 1 GB ram, micro SD card support for internal memory and whatnot. The number of ports and other facilities differs depending on the model of the Pi.

Pi runs on Raspbian open-source Debian based OS which uses Linux Distribution, so there is no need to spend to get the OS, unlike Windows. There are a couple of versions of Raspbian OS available including Jessie and Stretch. The desktop version is available with GUI making it more robust to use.

Python is the programming language of choice for Raspberry Pi. For the beginners who want to start software development on Pi, Raspberry Pi foundation recommends Python programming language It also supports Scratch, HTML5, JavaScript, jQuery, Java, C programming language, C++, Perl, and Erlang.

### 2.1.1 Raspberry Pi 3 Model B v1.2

Raspberry Pi model 3 is the first model of third-generation Pi SoC cards, first launched in 2016. In terms of the difference, it has a faster 1.2 GHz processor as compared to the previous versions (i.e., Pi and Pi 2) and also fused with built-in BLE and WLAN module. BLE serves critical energy efficiency while using the Bluetooth by switching off the radio and sending a low amount of data at low transfer speed. In terms of benchmark, the processor used in Pi 3 can deliver more than 50% performance enhancement dealing with 32 Bit application than Pi 2 which makes it Ten times faster than the first model of Pi. The third generation of Pi is first among the SoC cards which support 64 Bit applications. So that creates more opportunities in case of software development, especially for the beginners using Raspberry Pi Model 3.

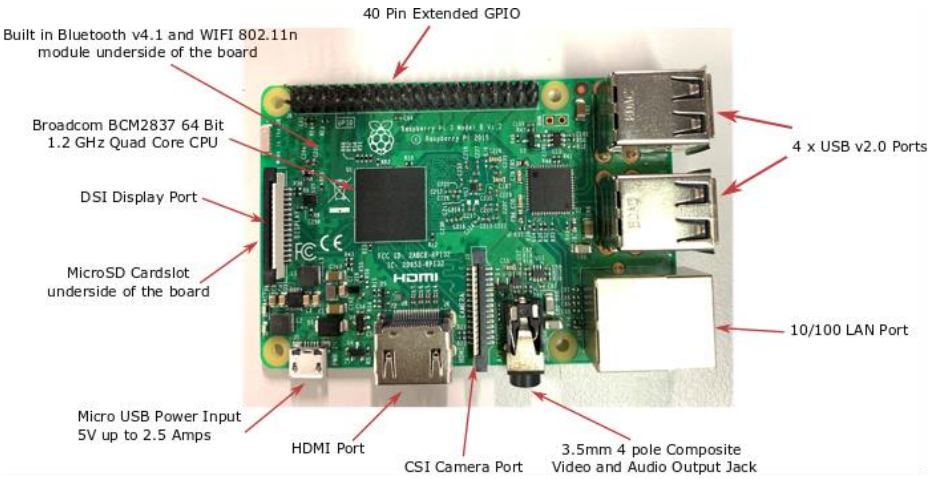



Figure 2-1 Raspberry Pi 3 Model B v1.2

Raspberry Pi 3 Model B v1.2 shown in Figure 2-1, supports more input current (up to 2.5 Amps) which enables us to power up more powerful peripherals through USB connection without using additional power source for the peripherals. Another mentionable difference is that it supports playing 1080p MP4 video at 60 FPS. Specification table is shown in Table 2-1. It will be in production till January of 2022.

| Model | Raspberry Pi 3 |
|---|---|
| Weight | 68 g |
| Dimension | 12.2 x 7.6 x 3.4 cm |
| Processor Brand | Quad Core ARM cortex-a53 |
| Processor Type | ARM 7100 |
| Processor Speed | 1.2 GHz |
| Processor Count | Quad (4) |
| RAM | 1 GB |
| Memory Technology | LPDDR2 |
| Graphics Coprocessor | Dual Core VideoCore IV |
| Graphics Chipset Brand | Broadcom |
| Graphics RAM Type | Shared |
| Graphics Card Interface | PCI-e |
| WLAN Type | 802.11bgn |
| USB Port Type | 2.0 (Total 4 Ports) |
| HDMI Ports | One available |
| Ethernet Ports | One available |
| Supported OS | Linux |

Table 2-1 Technical Specification of Raspberry Pi 3 Model B v1.2

Figure 2-2 Pin layout of Raspberry Pi 3 Model B

### 2.1.2 Ways to power up Raspberry Pi

The power supply of the Raspberry Pi plays a critical role in ensuring proper execution and operation. Because most of the cases, Pi becomes irresponsive as well as produce errors and crashes while running due to an inadequate power supply. It is recommended by Raspberry Pi to apply 5V and 2 Amp to satisfy the minimum requirement. The number of connected peripherals to the Pi increases the current it needs to perform stably. There are few ways available to power up the Pi. They are listed below.

1. Via micro USB port

   The easiest and recommended way to power up the Pi is to use the micro USB port available on the board shown in Figure 2-3. Directly connecting to a 5V input voltage and 2 Amps input current source will do the trick.

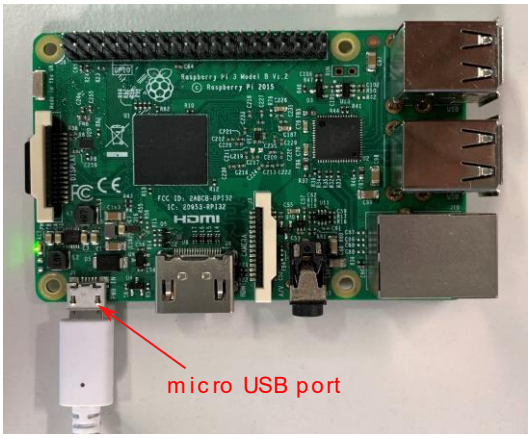



Figure 2-3 Powering up via micro USB port

2. Via GPIO port

   Another way to power up the Pi is by connecting 5V source input and Ground to the onboard GPIO pin shown in Figure 2-4. Depending on the model of the Pi numbering of the GPIO pins changes. In Pi 3 Model B, there are a couple of 5V and Ground pins available.

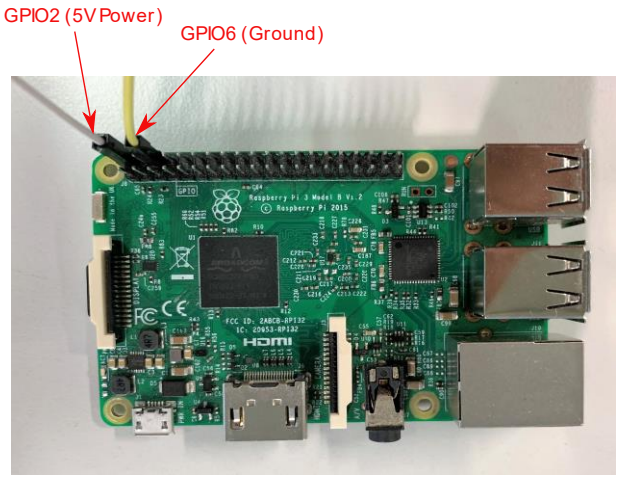



Figure 2-4 Powering up via GPIO port

3. Via USB port

   This way is kind of tricky. Pi is unable to be powered up directly by applying power on the USB port. Only after powering up the Pi through the micro USB port, Pi can run through the power of the connected USB cable.

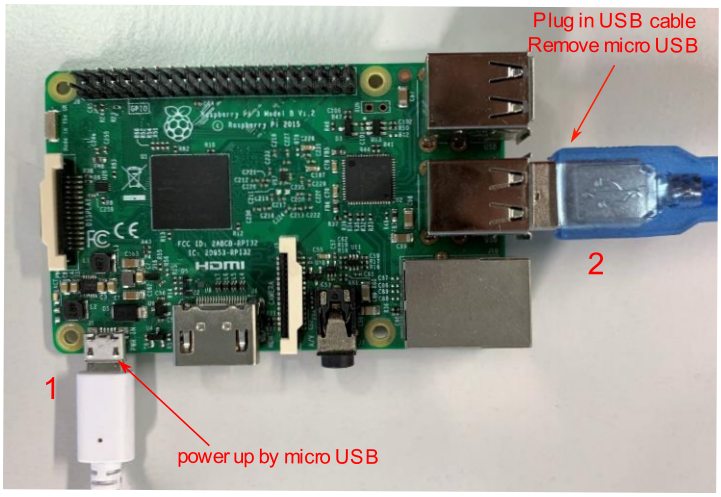



Figure 2-5 Powering up via USB port

That means, Pi needs to be booted up first via micro USB or GPIO ports (shown in Figure 2-5), then one can introduce the power through a USB port and remove the power from micro USB or GPIO ports.

The latest version of Raspberry Pi has a low voltage indication icon which indicates low voltage while running the OS. It is best to use the micro USB port to power up the Pi because it has inbuilt functionalities like fuse protection while powering up through the micro USB port. In the case of GPIO ports, there is no regulation or protection. Hence, one can end up permanently damaging the Pi or burning up the GPIO pins by applying an unregulated voltage and current through the GPIO ports. While keeping the Pi running through a USB port, one needs to keep in mind that the USB ports in Pi have a particular current limit of 500 mA.

Pi prone to create error while there is a fluctuation in input voltage, especially when the input voltage becomes less than 5V. Point to mention; other peripherals will take power from the USB ports of the Pi to power up themselves in case an establishment of a connection. It is suggested by the experts to provide input voltage 5.25V and current 2.5 Amp. It is better to use a little bit more than the 5V input voltage and 2 Amp current.

## 2.2. Arduino

Arduino is an open-source electric hardware and software. It designs and manufactures single-board microcontrollers or microcontroller kits. Depending on the model, a typical Arduino board can have many microprocessors and microcontrollers. It was first developed to provide use of the tool as Hardware and Software for fast prototyping for beginners in electronics and programming. Arduino has a community platform backed up by students, professionals, programmers, hobbyists around the globe. Thousands of projects and ideas using Arduino can be accessed freely on the web.

Arduino aims to provide an easy way for the young enthusiastic about creating devices that will interact with the environmental entities. By using the sensors like IR sensor, Ultrasonic sensor, RGB colour sensor attached with the Arduino, one can easily read data from the sensors writing a shortcode using Arduino IDE. The same settings go for actuators and motors as well. Motors such as DC, Servo, or Stepper can be attached to the Arduino, and a simple program written in Arduino IDE can control the motors.

There are also some advantage features worth to mention about Arduino. In terms of cost, Arduino microcontrollers are proportionately cheaper than other microcontrollers which are currently available in the market. Some Arduino modules are relatively cheaper than other Arduino modules as they need to be assembled by hand. Overall the high-end Arduino microcontrollers price is less than $50. Compatibility is also a significant issue that Arduino has managed to overcome. By providing Arduino IDE in Windows, macOS and Linux OS, where a majority of the microcontrollers follows on Windows platform.

Arduino's open-source IDE is very appealing for the beginners yet robust enough for advanced programmers. The environment follows Java programming language. An experienced user can also extend libraries using C++ on the IDE where the IDE follows AVR c programming language and open-source Softwares as all the things are open source to the users so the user can also check out the technical details which help a lot in the development.

Any person with adequate knowledge in circuit design can make his or her own Arduino module under Creative Commons License. That makes the hardware both open-source and extensible. Novice learners can also start up making their simple breadboard version module. That allows someone to learn and improve from scratch.

### 2.2.1 Arduino Uno

Arduino Uno is a microcontroller board shown in Figure 2-6. It has an ATmega328p microcontroller chip which is developed by Arduino.cc. The board has sets of digital (14 pins) as well as analog (6 pins) I/O pins. Six digital pins of the Arduino Uno also support PWM signals

| Microcontroller | ATmega328P |
|---|---|
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7V to 12 V |
| Input Voltage (limit) | 6V to 20 V |
| Digital I/O Pins | 14 (6 of them provides PWM output) |
| PWM Digital I/O Pins | 6 |
| Analog Input Pins | 6 (A0 to A5) |
| DC Current per I/O Pin | 20 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory size | 32 KB; 0.5 KB is being used by bootloader |
| SRAM | 2 KB |
| EEPROM | 1 KB |
| Clock Speed | 16 MHz |
| LED_BUILTIN | 13 |
| Length | 68.6 mm |
| Width | 53.4 mm |
| Weight | 25 g |

Table 2-2 Technical Specification of Arduino Uno

The board can operate by using an external power supply from 6 to 20 volts, and the recommended range is between 7V to 12 V (technical specification shown in Table 2-2). If the supply voltage is less than 7V the board may become

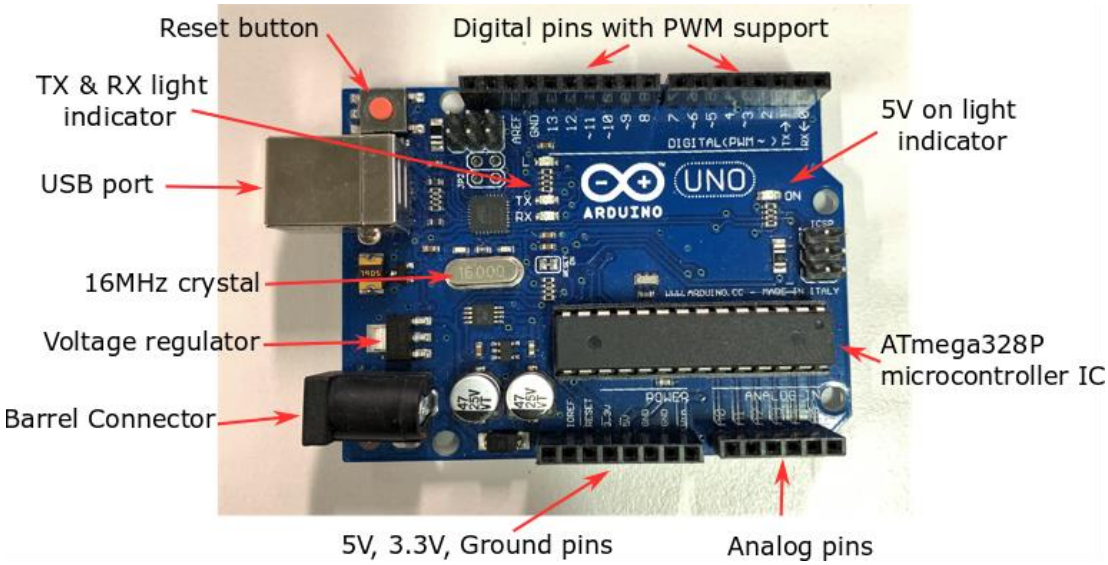



Figure 2-6 Arduino Uno Microcontroller Board

unstable and the 5V (provided by Arduino pin) levels on the board may fluctuate while using analogRead(). For this particular case, no problem has occurred by supplying around 5V to the Arduino Board through motor drive +5V output pin. In the case of supplying more than 12V, it can cause to board to overheat. Arduino Uno board is capable of providing 500 mA current in case of using a power source like a USB port. In the case of the use of an external power supply, the total maximum current draw will be 1Amp. The total maximum current per I/O pin is 40 mA, and the combined sum of currents getting out of all I/O pins is 200 mA.

### 2.2.2 Ways to power up an Arduino Uno

To power the Arduino Uno, one can either plug it into a USB port or introduce a voltage source to it either it is 2.1mm x 5.5mm DC power jack or via jumpers going to its "VIN" and "GND" pins. There is a total of four ways by which one can power up the Arduino Uno microcontroller. Which are as follows:

1. Using a USB cable

   It is the easiest way to power up the Arduino Uno board shown in Figure 2-7. The USB cable comes with the package of Arduino Uno. Arduino Uno can be turned on by connecting one end to the Arduino Uno board and another end to the USB port of a computer. Instead of that, one also can power up the Arduino Uno by using a power bank.

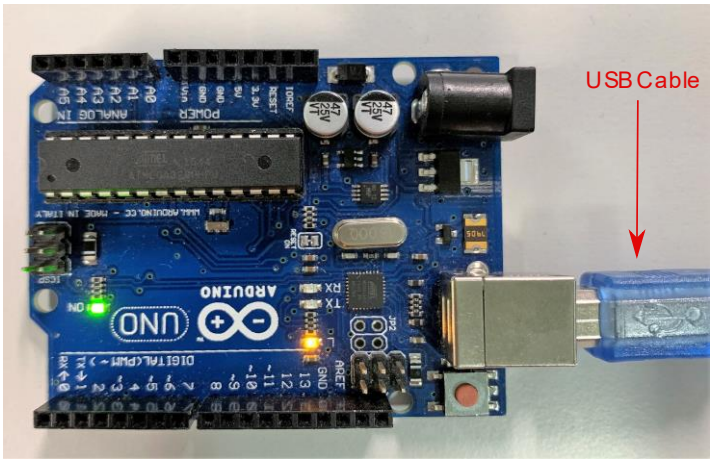



Figure 2-7 Powering up via USB port

2. Using an AC to DC adapter

   Another way to power up the Arduino Uno is to use AC to DC adapter. The adapter plug can enter into the barrel connector of the Arduino Uno board shown in Figure 2-8. The output voltage of the adapter can be in between 7V to 9V depending on the model of the adapter. Arduino Uno has an advantage over other microcontroller boards, as it has an onboard voltage regulator that regulates that excess voltage into a steady 5V.

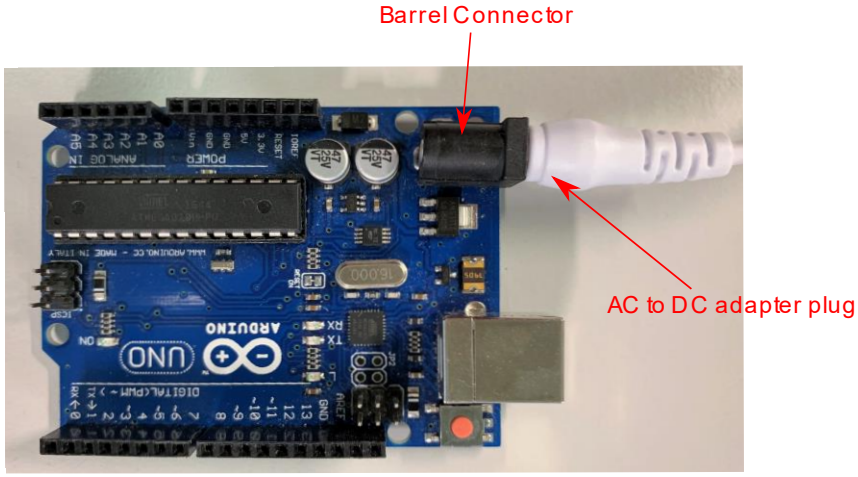



Figure 2-8 Powering up via Barrel connector

3. Using 5V input

Arduino Uno has onboard 5V input and ground pins. So, it can be powered up by applying 5V on the 5V input pin and making a common ground connection between the power source and the Arduino Uno board shown in Figure 2-9. As mentioned earlier, Arduino Uno has an onboard voltage regulator which saves the chip in case of fluctuation on voltage such as a sudden spike of voltage, which is higher than 5.5V. In any case, one can bypass that voltage regulator by applying 5V directly to the board. So, this procedure is somewhat riskier as one needs to maintain a constant and regulated input voltage.

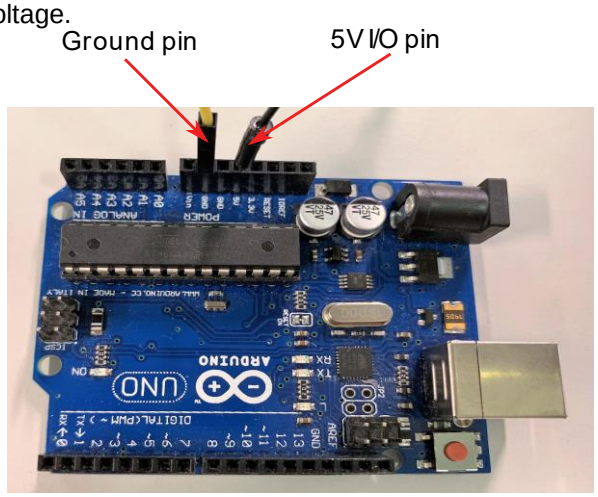



Figure 2-9 Powering up via 5V I/O

4. Using batteries greater than 5V connected in the $V_{in}$ pin of the Arduino Uno

In this case, Arduino Uno can be powered up by applying more than 5V on the Vin pin on the board. One has to maintain a common Ground shown in Figure 2-10. Input voltage on the Vin pin can be in between 7V to 12V.

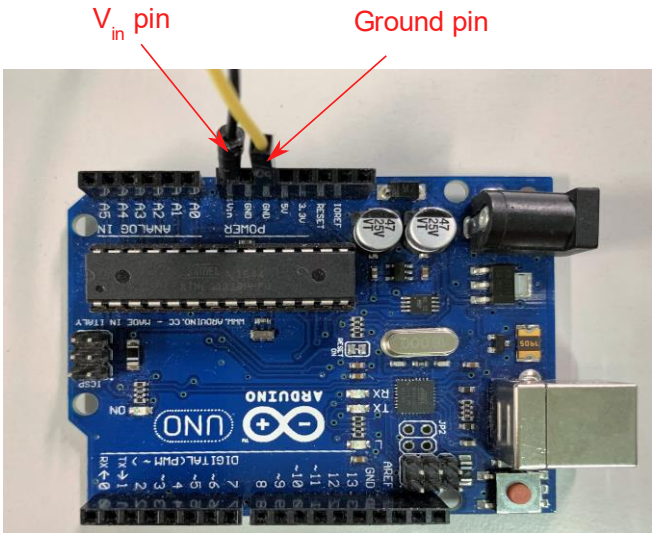



Figure 2-10 Powering up via $V_{in}$ pin

## 2.3. Battery

According to Eurobot2019 competition, one team can use only sealed batteries up to 12V and using the same battery one team has to perform three games in a row. This robot development uses two 6V Ultracell DC power supplies. One of the batteries is shown in Figure 2-11.


Figure 2-11 Ultracell VRLA 6V battery

Battery parameters are shown in the Table 2-3 below

| Model | Ultracell VRLA 6V |
|---|---|
| Part Number | UL4.5-6 |
| Length | 70 ± 2 mm (2.76 inches) |
| Width | 47 ± 2 mm (1.85 inches) |
| Container Height | 101 ± 2 mm (3.94inches) |
| Total Height (with terminal) | 106 ± 2 mm (4.17 inches) |
| Approx. Weight | Approx. 0.75kg (1.65lbs) |

Table 2-3 Battery Specification Ultracell VRLA 6V

Initially, one can use motors rated as 6V, but the other circuit modules support 5V voltage input such as Arduino, Raspberry Pi. Also, one needs to use a DC to DC converter to step down the voltage. The converter stabilizes the voltage so that the circuitry works properly.

## 2.4. DC Motors

DC motor produces mechanical energy by using the electrical mechanisms inside it while taking the electrical energy as input. The thesis follows two chr-gm25-370 6V motors shown in Figure 2-12 to run the robot.

Figure 2-12 chr-gm25-370 6V motors

These are magnetic Holzer encoder gear motor with protective Shell made by the company CHIHAI MOTOR made by China.

Technical Specification of the motor is shown in Table 2-4 below

| | |
|---|---|
| Type | AB dual phase incremental encoder |
| Pulse outputs of rotation | Basic pulse 11 PPRxgear reduction ratio |
| Holzer power supply voltage | DC 3.3V/DC 5.0V |
| Basic function | With the pull-up shaping resistor, a single chip microcomputer can be used |
| Interface type | PH 2.0-6 PIN (providing wiring) |
| Output signal type | Square wave AB phase |
| Basic pulse number | 11 PPR |
| Number of triggers pulse of magnetic ring | 22 poles (11 pairs of poles) |
| Revolutions Per Minute (RPM) | 600 RPM |
| Working current | 0.2A |
| Working temperature | -20 to 60 ℃ |
| Rate voltage | 6V |
| Power range | 3V~6V |
| Input voltage | DC 6V |
| Material used | ABS, zinc alloy, steel, copper, nylon |

Table 2-4 Technical Specification of chr-gm25-370 6V DC motor

# 2.5. Servo Motors

Four servo motors attached in the mechanical arm of the robot. There is one LDX-218 digital servo (technical specification shown in Table 2-6), two LFD-06 anti-blocking servos (technical specification shown in Table 2-6), one LDX-335 MG digital servo (technical specification shown in Table 2-7), and one LDX-335MG anti-blocking servo. Servo motor has geared DC motor in it with following the closed-loop circuit. The motor can move its output shaft in a precious angle by using the motor controller and position encoder. The motor controller and position encoder give freedom for the user to apply precious angular placement as well as velocity and acceleration. By feeding different electric signals to the moto the angular position of the servo changes.

| | |
|---|---|
| Brand | LOBOT |
| Model | LDX-218 |
| Operating Speed (7.4V) | 0.16 sec/60° |

| | |
|---|---|
| Stall Torque (6.6V)<br>Stall Torque (7.4V) | 15 Kg-cm<br>17 Kg-cm |
| Dimensions | 40 x 20 x 40.5mm |
| Weight | 60g |

Table 2-5 Specification LDX-218

| | |
|---|---|
| Brand | LOBOT |
| Model | LFD-06 |
| Voltage | 6.9V to 7.4V |
| Dimensions | 40 x 20 x 40.5mm |
| Weight | 60g |

Table 2-6 Specification LFD-06

| | |
|---|---|
| Brand | LOBOT |
| Model | LDX-335MG |
| Voltage | 6.9V to 7.4V |
| Dimensions | 40 x 20 x 40.5mm |
| Weight | 60g |
| Stall Torque (6V)<br>Stall Torque (7.4V) | 15 Kg-cm<br>19 Kg-cm |
| Working current | 100mA |
| Operating speed (7.4V) | 0.16 sec/60° |

Table 2-7 Specification LDX-335 MG

Power, Ground, and control are the three input pins of a conventional servo motor. There is an initial position for all the servo motors. Whenever a control signal reaches the motor, it changes its degree and maintains the position. The control pin is responsible for feeding the signal to the servo motor where the control pin is usually connected to the microcontroller I/O pins.

# 2.6. LM2596 Buck Converter

LM2596 is a DC-DC step-down PSU with voltage meter (shown in Figure 2-13) with output voltage 1.3V-35V and maximum output current 2A. It is a step-down converter which can step down voltage while a step up the current.

The LM2596 DC-DC step-down converter has integrated digital voltage meter support. Pressing the switch button one can toggle the display showing input and output voltage. The digital voltage meter is capable of showing both input and output voltage. The converter also has an onboard multi-turn potentiometer. One can apply the input voltage from a DC source starting from 4V to 35V and can adjust the voltage output from 1.3V to 35V (technical specification is shown in Table 2-8). Another additional advantage of using this converter is that it supports up to 2A current.

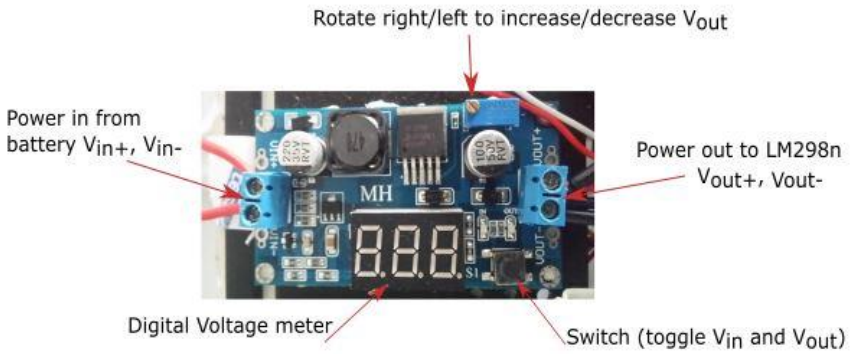



Figure 2-13 LM2956 Buck converter

| | |
|---|---|
| Regulator Type | Step Down |
| Input Voltage | +4 to 40vdc |
| Output Voltage | +1.25 to 35vdc |
| Output current | 2A (3A maximum with heat sink) |
| Switching Frequency | 150kHz |
| Efficiency | Up to 92% (when the output voltage is set high) |
| Dropout Voltage | 2vdc minimum |

Table 2-8 Technical Specification of the LM2596 buck converter

From the battery, one can wire the positive terminal to the Vin+ and negative terminal to the Vin- of the LM2596 buck converter. From Figure 2. 4, one can observe the main components of the converter. Rotating the screw left and right one can increase and decrease the output voltage from 4V to 35V. Digital voltage meter showing the current input/output voltage. One can toggle the voltage reading by pressing the switch S1.

### 2.6.1 Limitations

This module can handle 1.5A properly where it can handle 2A for a short period under the condition that the difference between the output and input voltage is short.

## 2.7. L298n DC Motor Drive

It is a dual H-Bridge motor drive that can control the speed as well as the direction of the motor shown in Figure 2-14. DC motor voltage is 5V-35V, and the peak current is 2A. There are two ways to control the speed of the motor using the motor drive

- Higher input voltage produces higher DC motor speed.
- Adjust the speed through PWM by connecting to the Arduino's PWM output pins.

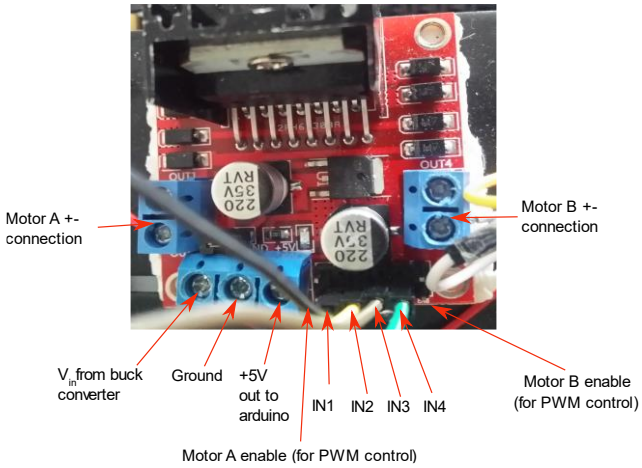


Figure 2-14 L298n module

General specification of L298n motor drive is given in the following Table 2-9

| Motor driver | L298N |
|---|---|
| Motor channels | 2 |
| Maximum operating voltage | 46V |
| Peak output current per channel | 2A |
| Minimum logic voltage | 4.5V |
| Maximum logic voltage | 7V |
| Package | Multiwatt15 |

Table 2-9 Specification of the L298n motor drive

Figure 2.5 is showing the connections of the L298n module. On the left side, there are positive and negative pins of motor A, and on the right side, there are positive and negative pins of motor B. The +5V output of the motor drive to power up the Arduino board or other microcontrollers where the voltage is convenient. IN1, IN2 are used to control the motor A rotation, and IN3, IN4 are used to control the motor B rotation. By using the command HIGH, LOW or LOW, HIGH and LOW, LOW one can give the direction of the rotation forward, backward and stop respectively. To control the motor speed by using PWM (Pulse Width Modulation), we are removing the jumpers both right and left of the IN1, IN2, IN3, and IN4 pins. Stating the enA and enB value, one can control the speed of motor A and motor B. The value of enA and enB can be set from 0 to 255.

## 2.8. DC to 5V USB converter

Basically, this is a mini car charger produced by RAVPower, which converts the DC voltage to 5V steady voltage shown in Figure 2-15. It has constant loading technology that regulates the input voltage changes

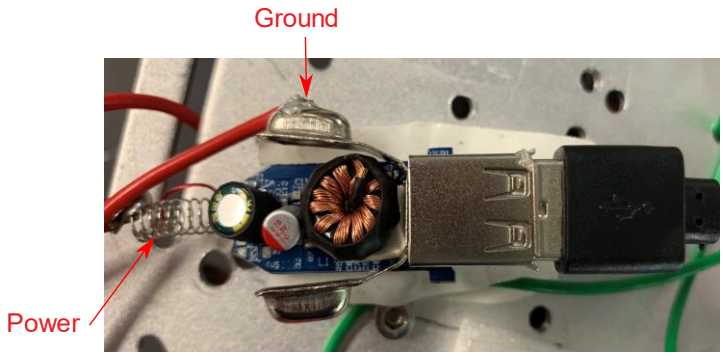


Figure 2-15 DC to 5V USB converter

which minimizes over-voltage and under-voltage problem. There are in total two USB ports available capable of delivering 4.8A output current.

## 2.9. Raspberry Pi Camera

Manufactured by Raspberry Pi, this camera module can be attached to the Pi board by flex cables. Depending on the necessity, various length of flex cable is available on the market. A 150 mm (approx. 6 inches) flex cable comes by default in the packaging when a Pi camera module is purchased. The price range is around $35.

Figure 2-16 Raspberry Pi Camera v2.1

The Pi camera can shoot 1080p photos and videos while it can be controlled by writing program script in Python programming language shown in Figure 2-16. The following Table 2-10 is showing the specification of a Pi camera:

| Name | Raspberry Pi Camera Module |
|---|---|
| Sensor | 5 megapixels with a fixed focus lens |
| Image Resolution | Supported up to 2592 x 1944 pixels |
| Video Resolution | For 30 FPS 1920 x 1080<br>For 60 FPS 1280 x 720<br>For 60/90 FPS 640 x 480 |
| Price | $35 |

Table 2-10 Specification of Raspberry Pi Camera

There are some pre-defined functions to work with the Pi camera. Some of them are PiCamera(), start_preview(), stop_preview() and capture() shown in the Table 2-11 below.

| | |
|---|---|
| PiCamera() | For initialization of Pi Camera |
| start_preview() | Start viewing live video stream |
| stop_preview() | Stop viewing live video stream |
| capture() | Capture Image |

Table 2-11 Some Pi camera functions

# 2.10. Communication between Pi and Arduino

## 2.10.1 Ways of Communication

This thesis deals with Raspberry Pi and Arduino Uno. Raspberry Pi does object detection. Arduino Uno is being used for the movement of the DC and Servo motors of the Robot Chassis and Mechanical Arm respectively. As both of them are doing different tasks, there has to be a link so that Raspberry Pi will be detecting the object and will be sending messages to Arduino for navigation. There are three different ways to establish communication between Pi and Arduino. They are

1. USB

   One can establish a communication between Pi and Arduino by using the USB cable provided for the Arduino. By merely putting one end on the Pi's USB port and another end on Arduino's USB port.

2. Serial communication

   This communication uses Serial Peripheral Interface, which is a synchronous serial bus system. Establishing a serial communication is kind of complex as it needs four wires to make a connection between one master and one slave. So, in the case of multiple slave architecture, the number of wires will be high.

3. $I^2C$ communication

   I-squared-C is the most preferred communication medium. It requires only two wires to establish a connection between one master and up to 1008 slaves. It also supports multi-master salves architecture. This thesis follows $I^2C$ communication which will be discussed in the further chapter.

## 2.10.2 $I^2C$ Bus Communication

I2C stands for Inter-Integrated Circuit where the bus communication can facilitate double direction communication. It is a kind of a protocol by which one can connect two Integrated Circuits. It is a bus type of communication where each device connected to the bus has its unique address. I2C bus communication is suitable for low-speed applications where the speed is limited to 10kbs. Important to mention that recent developments offer to speed up to 3.4Mbps.

Only a few wires are needed to make a connection between Pi and Arduino using I2C bus communication shown in Figure 2-17. I2C communication uses pins A4 and A5 of the Arduino where SCL1 and SDA1 pins of the Raspberry Pi. In this communication, the Ground of both Pi and Arduino Uno must remain in the same connection.

SDA stands for Serial Data line, and SCL stands for Serial Clock line. The Serial Data line carries the data on the bus line of the I2C communication protocol. Serial Clock is generated from master only where it is used to maintain the synchronization on the bus use.

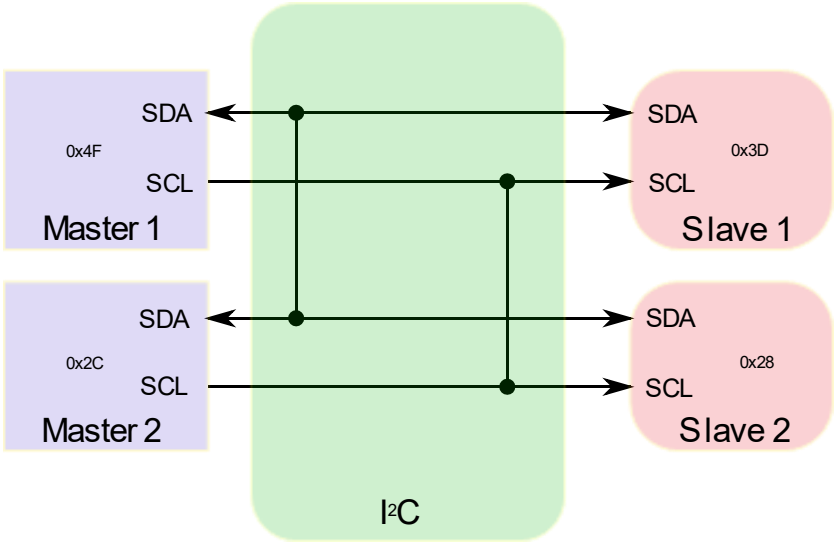



Figure 2-17 I²C in multiple master-slave architecture

In the figure, one can see two masters and slaves having address 0x4,0x2C, 0x3D and 0x28, respectively. SDA supports bi-directional data transfer between master and slave, where SCL has unidirectional transfer support. The SCL is maintaining the clock synchronization for the usage of the bus line of the I2C protocol. There are three types of I2C operations. They are

1. Master writes to the slave device.
2. Master only reads from the slave device.
3. Master writes in the slave device and reads from the slave device.

One of the main advantages of I2C communication is Flexibility as it supports multi-master and multi-slave communication. Another one is the addressing feature as each device connected to the bus has its unique address. So, one can add more components to the bus at ease. Simplicity fact can also be counted as one advantage as fewer wires are needed to connect two Arduinos.

I2C bus communication protocol uses ACK/NACK (acknowledgement/no acknowledgement) as an error handling mechanism. Adaptability is also an advantage of using the I2C bus communication protocol as it can work in both fast and slow Integrated Circuits.

There are very few limitations of the I2C bus communication protocol. Not to forget this protocol is running for 30+ years. Address conflict might occur in multiple masters and many slave configurations. Speed is limited to the peers of I2C bus communication as it uses pull-up resistors, not the push-pull resistors. It also requires significantly more space as it uses pull-up resistors.

## 2.11. TensorFlow

Google Brain Team has developed TensorFlow, which is a software library. It makes conventional machine learning quicker as it uses neural networks for the deep learning process. It focuses on numerical computation and large-scale machine learning. Google is currently using TensorFlow in both research and production.

TensorFlow is Python friendly and uses it as front-end API in case of creating applications with the framework. It uses C++ to run those applications for its high performance. Here Python is being used to achieve high-end programming abstraction. The general algorithm of TensorFlow in shown in the Figure 2-18 below

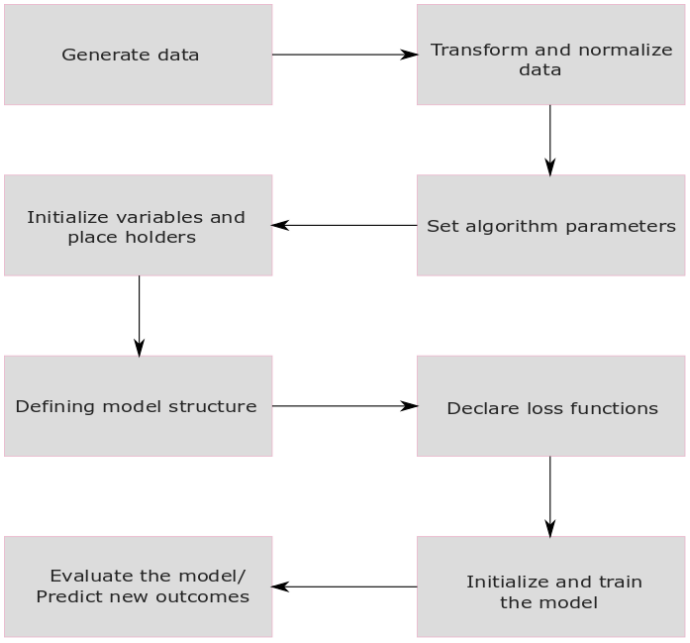



Figure 2-18 General Algorithm of TensorFlow

The dataflow graph describes how the data is flowing through the graph. TensorFlow lets the user generate that Dataflow graph using scientific computing and deep learning algorithms. Here the graph is treated as nodes. Each node describes some operations, and the edges connecting the nodes are called tensors. Nodes and tensors together form Python objects.

## 2.12. Operating system

The thesis follows three operating systems in total. Linux distributed Debian OS is for running the Raspberry Pi. macOS is for coding in the Arduino and Raspberry Pi. Windows 10 is used to train the object detection classifier and to create a frozen dataflow graph for TensorFlow. An abstract of OS hierarchy is shown in the Figure 2-19 below.

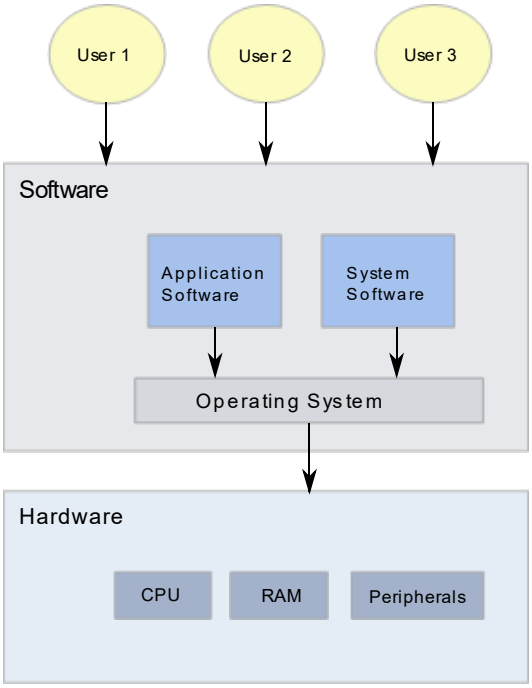


Figure 2-19 Operating System hierarchy

The operating system plays a vital role to control all the peripherals connected to a computer. It is the system software that establishes the bridge between peripherals and other programs running on the OS. It creates an environment for the user and the programs and other peripherals where users can access the program while using peripherals and other software and hardware resources. Figure 2-20 showing the functions of the Operating System.

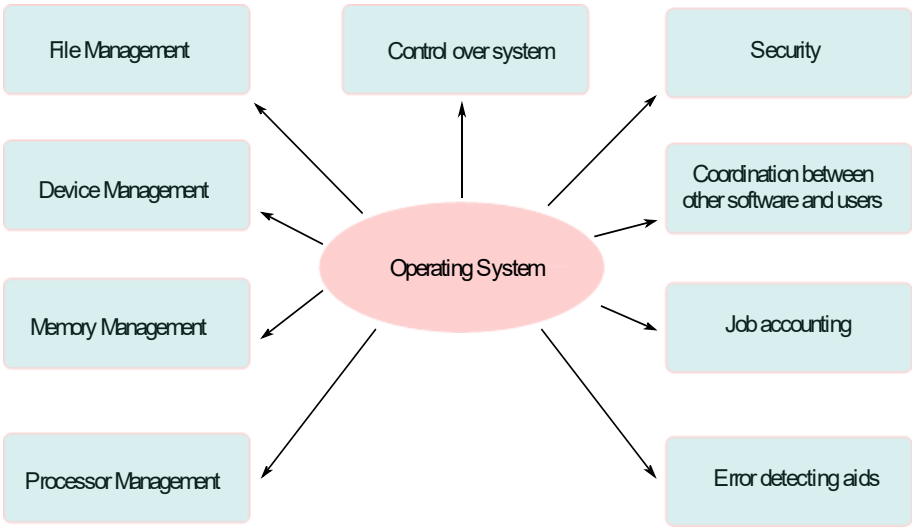


Figure 2-20 Functions of the Operating System

## 2.12.1 Debian OS

Debian is the OS for Raspberry Pi. Many programming enthusiasts have made numerous Debian Projects eventually forming the Debian OS. This OS is completely free to use. With a proper internet connection, one can easily download this OS. A proper internet connection is essential here as the OS file is usually significantly large. In case of installing the OS using CD, a user has to pay the price for the CD other value-added costs.

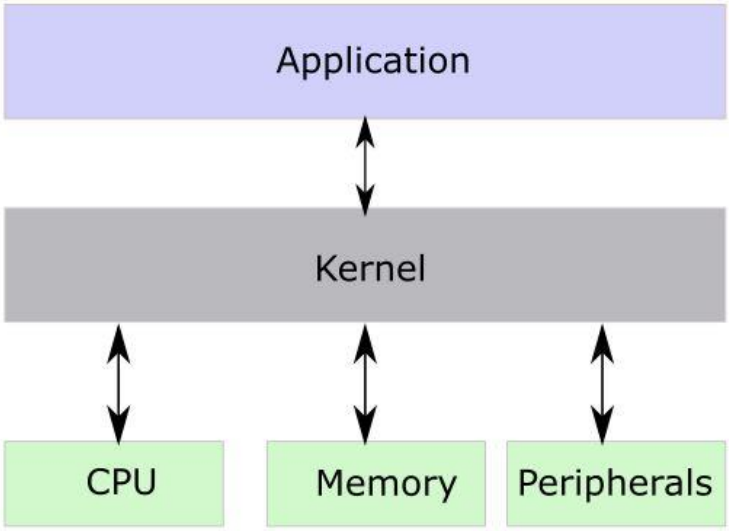



Figure 2-21 Kernel making a bridge

Debian OS runs on Linux kernel. Linux is an OS that is also free and supported by thousands of programmers worldwide, and the kernel is a program on the OS which does all the necessary works (shown in Figure 2-21 Kernel acting as the bridge) and acts as the leading role to start other programs on the OS. The kernel is the kind of the heart of the OS which has full control over the system.

### 2.12.2 Windows 10 & MacOS

This thesis follows various applications and software. Some of the software and applications are not that much supported or compatible with Windows 10 or macOS. So, we have used both.

TensorFlow trains the object detection classifier. This part is done using Windows 10 OS and CPU. In the case of software development, the thesis follows the usage of macOS. macOS has particular advantages over Windows 10 in terms of software development. As in macOS, it takes much less time to compile and run a program. Even uploading a code through Arduino IDE into Arduino using Windows 10 takes more time than macOS. The implementation section covers the software development part of the thesis.

## 2.13. Programming Languages

To operate on almost any hardware device with control options, we need a programming language. The programming language will contain the set of instructions addressing the functionality of the hardware device. In other words, programming language consists of all the vocabulary and rules and regulations to perform a specific task on a hardware device. To work on a programming language, we need to sync ourselves with the syntax and semantics of the programming language.

Some hardware devices support several programming languages. In some cases, the developers' development themselves preferable programming language support environment for the unsupported hardware devices.

### 2.13.1 C/C++

Arduino Uno microcontroller supports C/C++ software development. Arduino language, which is a set of C/C++ functions, has preprocessing facilities. So, in many function calls, it is not required to do the function prototyping.

### 2.13.2 Python

This thesis focuses on Python programming in the software development of the Raspberry Pi.

## 2.14. Softwares

### 2.14.1 VNC Viewer

VNC stands for Virtual Network Computing. VNC Viewer provides the functionality to share the desktop among computers like Pi to another computer or laptop. It uses the RFB protocol to share the desktop between two different entities. The following Figure 2-22 shows the overview of the VNC Viewer

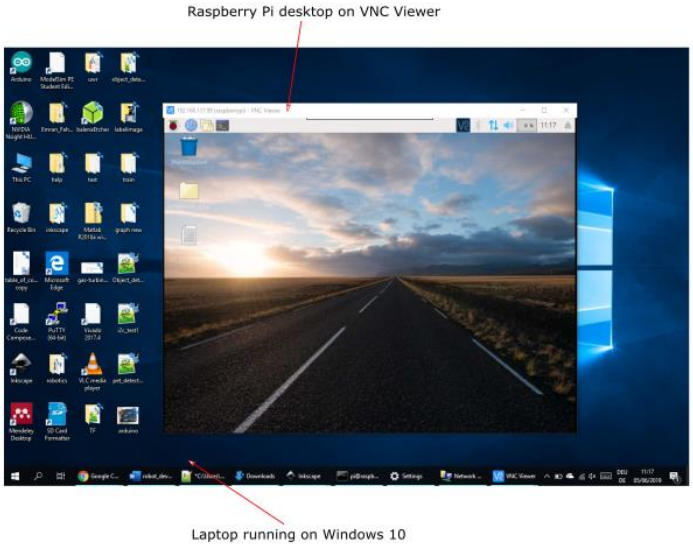



Figure 2-22 VNC Viewer overview

In this protocol, mouse events and keyboard events are stored as graphical screen updates and then shared between the entities.

### 2.14.2 LabelImage

It is a tool that provides a graphical image annotation facility. LabelImage follows Python programming. The graphical interface of LabelImage follows Qt. An overview of LabelImage in shown in the Figure 2-23 below.

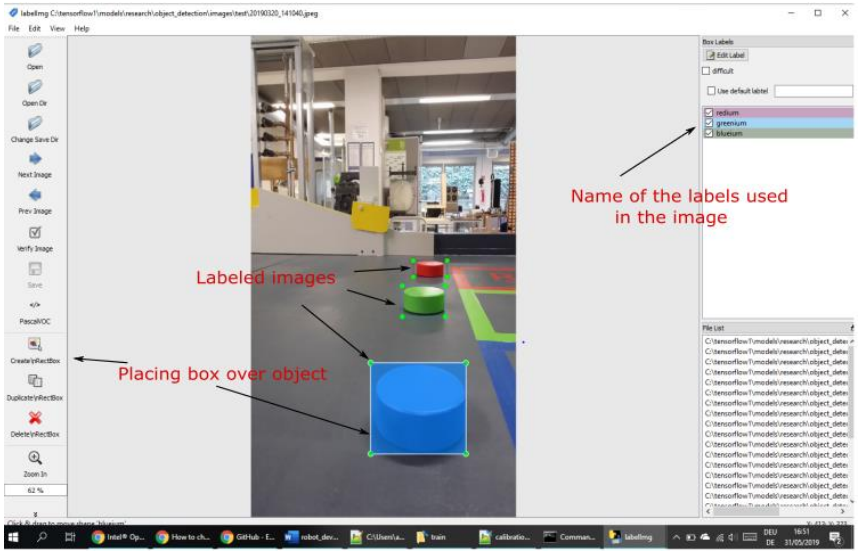


Figure 2-23 LabelImage overview

The image annotation procedure follows by placing a bounding box upon the particular object of the image. It generates an XML file for the image in the PASCAL VOC format.

## 2.14.3 Anaconda (Python Distribution)

Anaconda Python Distribution provides scientific computing features for Python programming language. Scientific computing features consist of data science, large scale data processing, and predictive analytics for various machine learning applications.

Anaconda Distribution consists of more than 1500 packages and also has a virtual environment manager. Virtual environment manager provides a whole range of benefits for managing multiple Conda environment. Conda acts the role, such as analyzing the current environment and providing necessary limitation information. Conda is also responsible for managing libraries and dependencies among the python packages.

Anaconda command prompt will be available to use after installing Anaconda Python Distribution. It is similar to other command prompts such as Windows command prompt or macOS's terminal command prompt shown in Figure 2-24.

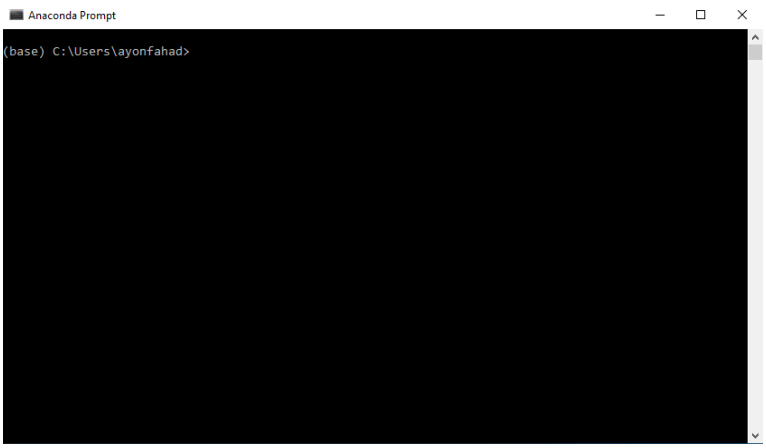


Figure 2-24 Typical Anaconda Command Prompt

Anaconda command prompt gives the user the flexibility to start writing Conda commands without fixing the directories or path to use Anaconda.

# Chapter 3 Eurobot 2019

## 3.1. Introduction

It is an International Robotic Contest for the nonprofessional people aged under 30 years. After competing at the National level, students from all around the globe compete with each other. Usually, a couple of students together form a group, and two groups compete with each other in per game.

Eurobot encourages students to invest their technical knowledge in a practical form. It follows as an active learning process where the students enjoy doing their experiments. Students have to build a robot/s from scratch. A supervisor is permissible to guide each group where the supervisor cannot actively work on the student project but only convey advise and guide the students. The group of students has the option to develop up to two autonomous robots where the second robot is optional.

Formed in France, Eurobot Finals are always taking place in Europe. The team has to be registered into Eurobot to compete in this platform. When there become more than three teams at the National level, a National qualification occurs. The national qualifications also take place in Europe. Every year the number of participation of teams is increasing. In recent year 2018, around 1500 students from 15 countries complete each other in Eurobot. Most of the cases, France takes the first prize of Eurobot in number 19 times out of 21.

## 3.2. Atom Factory

Every year the rules and theme of the Eurobot changes. In 2019, the theme of Eurobot is Atom Factory. The idea of the theme came from Dmitri Mendeleev's discovery of the Periodic law and the Periodic table. He constructed and published a table of known elements by understanding the periodic pattern. He discovered by focusing on the atomic mass elements can be classified. He was also successful in predicting the atomic mass and properties of the missing elements at that time. He placed the elements showing similar properties in a vertical order. The elements were serialized based on their atomic mass.

### 3.2.1 Playing element

There are four playing elements in the Atom Factory shown in Figure 3-1. They are called atoms. In this case, atoms are ice hockey pucks. The colour and weight of the pucks are different shown on the Table 3-1. In other words, pucks are identical to each other, having different outer colour and mass.

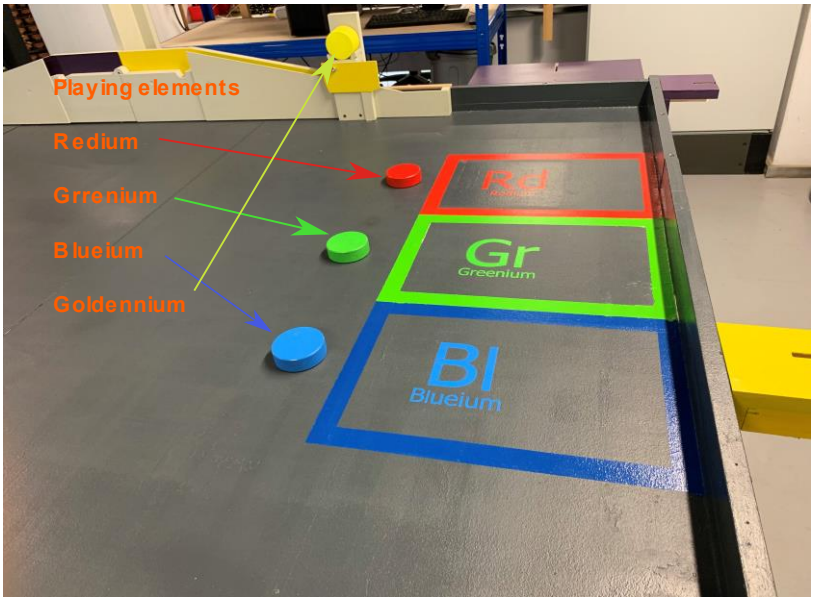



Figure 3-1 Playing elements

| Atom Name | Type | Description | Weigh |
|---|---|---|---|
| Redium | Standard | Very common, light | 60g |
| Greenium | Standard | Common, little heavier | 120g |
| Blueium | Standard | Rare, rather heavy | 170g |
| Goldenium | Special | Extremely rare, heavy | 340g |

Table 3-1 Playing elements specification

### 3.2.2 Starting areas

In the playing area, there is one starting area for each team. In each game, two teams will compete with each other. Each team will get a place consisting of three periodic tables, which are Redium. Greenium and Blueium. The starting area mainly consists of two cells of the periodic table Redium and Greenium. That means the robot has to be placed within the Redium and Greenium periodic table when the game starts. We have to develop a robot that can be placed within the starting area properly and also keep in mind that it is taking the allocated area only.

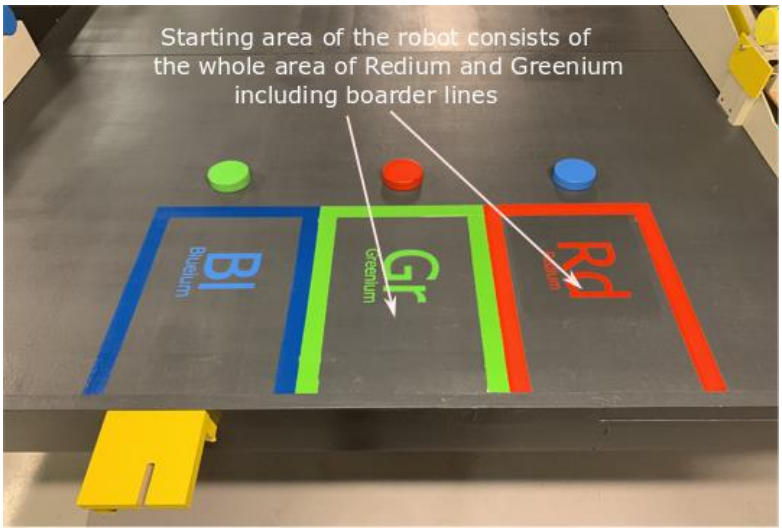



Figure 3-2 Starting area

From the Figure 3-2, one can see the permitted starting area of the Robot(s) which consists of Greenium and Redium cell of the periodic table.

### 3.2.3 Playing area

The playing area is the area where two teams compete with their robots shown in Figure 3-3. It consists of eleven different sub-areas. As a whole, it is rectangular like a horizontal surface.

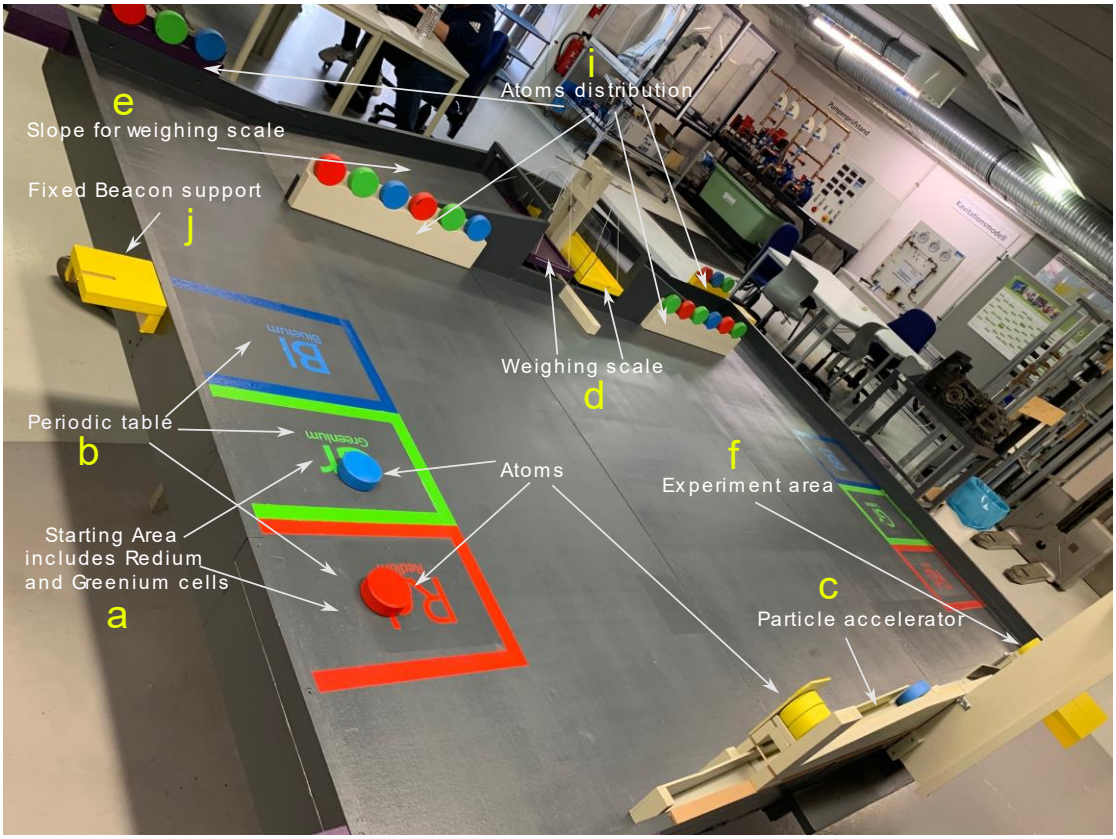



Figure 3-3 Playing area

a. Starting areas
b. Periodic table
c. Particle accelerator
d. Weighing scale
e. Access slope for the weighing scale
f. Experiment area
g. Chaos area
h. Oxygen atom
i. Atom distributors
j. Fixed beacons support
k. Central tracking device

In the Figure 3-3, one can notice that some of the playing elements in the playing area are missing. These elements are not currently available in the playing area, which this thesis follows. Nevertheless, for the completing first mission every elemets are available.

## 3.3. Missions

In Eurobot 2019, tasks of the robot are doing some experiments autonomously. They are listed as follows

1. Classifying the atoms.
2. Weighing the atoms.
3. Creating a new element.
4. Do an experiment.
5. Predict the unknown elements.

### 3.3.1 Classifying the atoms

In this mission, robot(s) has to classify atoms according to their weight or colour and put them in their corresponding periodic table. Three atoms, named as Redium, Blueium, and Greenium, will be placed arbitrarily in front of the periodic table shown in Figure 3-4.

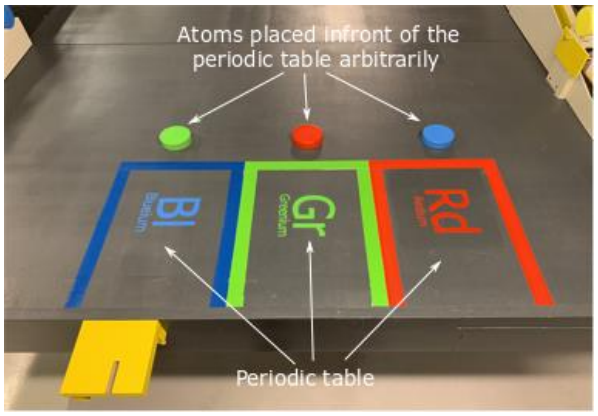



Figure 3-4 Atoms placed in arbitrary order

The robot has to classify the atoms by putting them in their periodic table. As shown in the Figure 3-5, the red, green, and blue cells are the allocated area for classifying the atoms of Redium, Greenium, and Blueium.

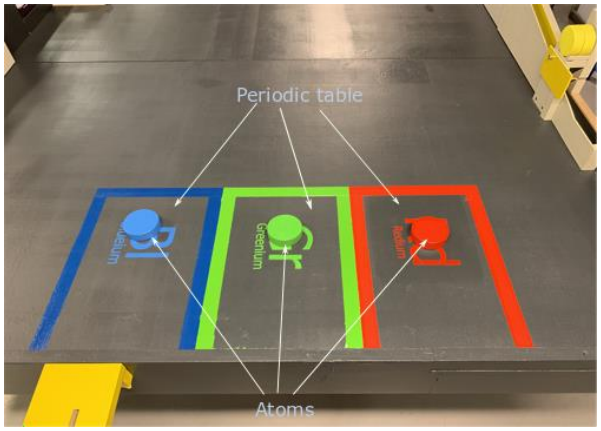



Figure 3-5 Atoms placed in the respective periodic table

The Figure 3-5 also shows that all periodic tables contain the appropriate atoms. As the playing area is built for two teams competing for each other, so the robot(s) of the respective team has to classify the atoms within their periodic table cells. There are also a few other constraints to follow

1. Atom's vertical projection has to be on the playing area or another atom. It also needs to be partially on any cell of the periodic table.
2. There are points for putting atoms in any cell. If the atoms are put partially on the corresponding cell, the action will make more points.
3. Atom Goldenium can be placed on any cell of the periodic table to earn extra points.
4. It is not allowed for the robots to enter into the opponents periodic table neither hamper with the atoms placed on their cells between the starting and ending time of the match.

### 3.3.2 Weighing the atoms

There is a weighing place in of the playing area where there are two weighing trays one for each team shown in Figure 3-6. The weighing scale follows the Double Pan Balance Scale structure, where it can move up and down. It stays a horizontal position when both of the trays are empty.

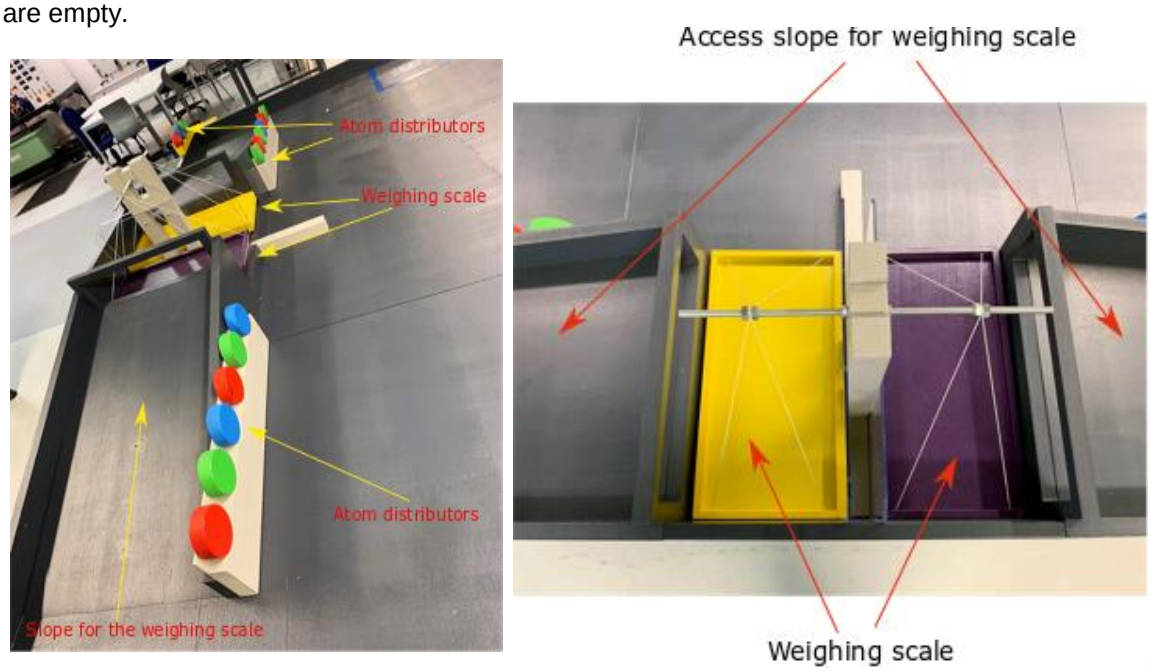



Figure 3-6 Weighing Scale

Robots of each team will be picking up the atoms and putting them in the weighing scale. According to the weight of the atoms, each team gets its point.

### 3.3.3 Creating a new element

In this mission, each team has to use put an atom on the particle acceleration shown in Figure 3-7. As there is a ramp, the atom will follow the bottom of the ramp.

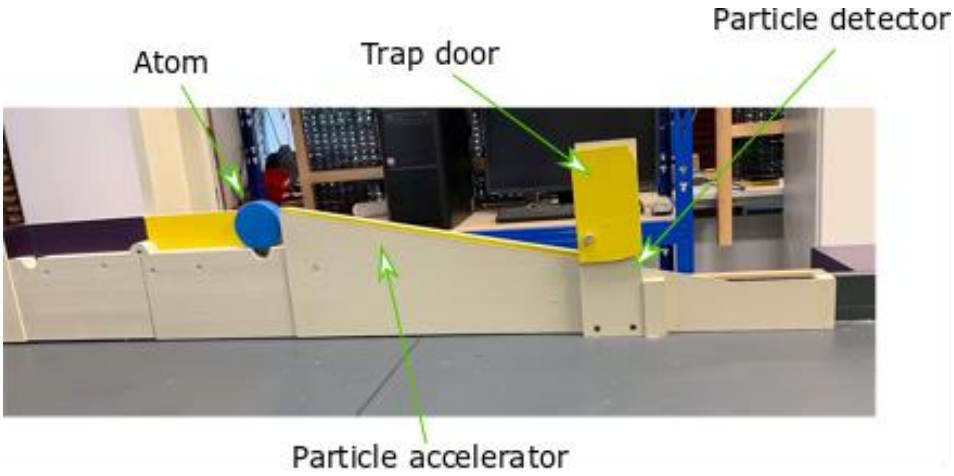



Figure 3-7 Creating a new element demonstration 1

When the atom reaches the bottom of the ramp, it touches a trap door. The particle detector is that door that opens and releases Goldenium shown in Figure 3-8

.

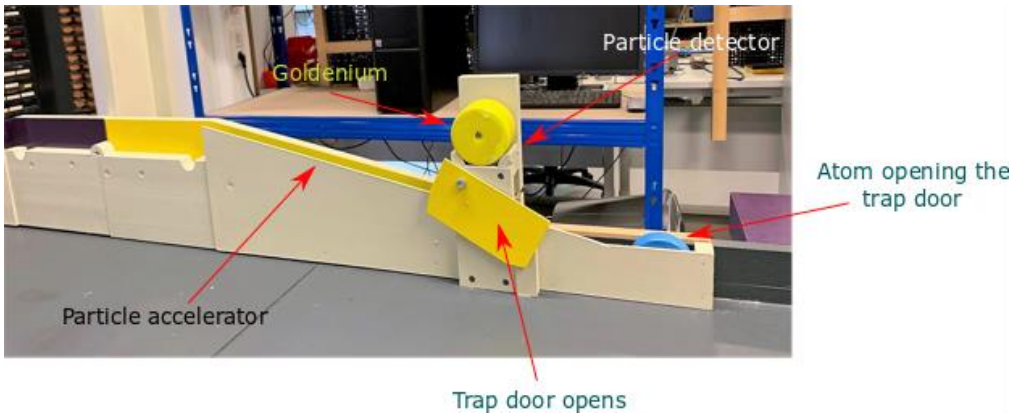



Figure 3-8 Creating a new element demonstration 2

### 3.3.4 Do an experiment

Own experiment consists of doing some tasks using the experiment area (shown in Figure 3-9), oxygen atom, ionic bonding, the experiment, and the election. Each team has to do their experiment. They have to develop a way to move the electron from the experiment area to the Oxygen atom area. A short overview of the mission is provided below.

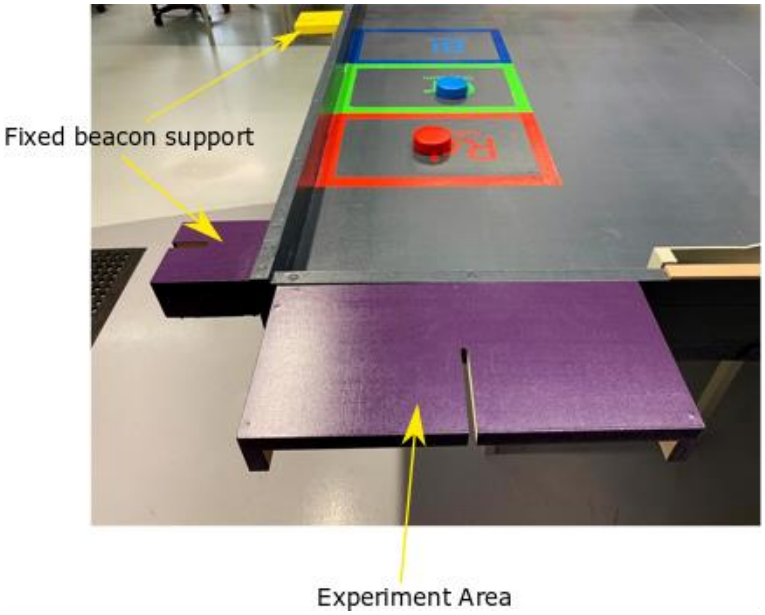



Figure 3-9 Experiment Area

1. The position of the Experiment area is on the corners near the playing area. Its height is adjusted matching the height of the board of the playing area.
2. The oxygen atom is a decorative part designed by the organizers.
3. Ionic bonding is a cord. Each playing team is open to use their own made cord.
4. The electron has to be moved from the experiment area to the oxygen atom area using the Ionic bonding.

### 3.3.5 Predict the unknown elements

In this mission, each team predicts the score their robots will obtain during the match and puts it on a paper or electronic display. In the case of using a paper, the teams have to write their pre-evaluation scores in decimal numbers. The use of dynamic display will earn bonus points for the team where teams can develop a dynamic score evaluation during the match using an electronic display. Teams can place their electronic display on the robot(s) or the experiment area of each team.

# Chapter 4 Implementation

## 4.1. General Strategy

As discussed earlier, this thesis paper focuses on completing the first mission of Eurobot 2019, which is classifying the atoms. The general strategy of the implementation process is as simple as water that is developing a robot performing the desired mission.

To simplify the mission, we have divided the mission into a couple of submissions. We focused on the submissions and got to know about many insights. Submissions are provided below.

1. The robot detects the atom using Pi Camera.
2. The robot moves forward, and Mechanical arm grabs and pulls the atom up.
3. The robot moves to the respective periodic table of the atom.
4. Mechanical arm pulls down the atom on the corresponding periodic table.
5. The robot gets ready to detect the next atom.

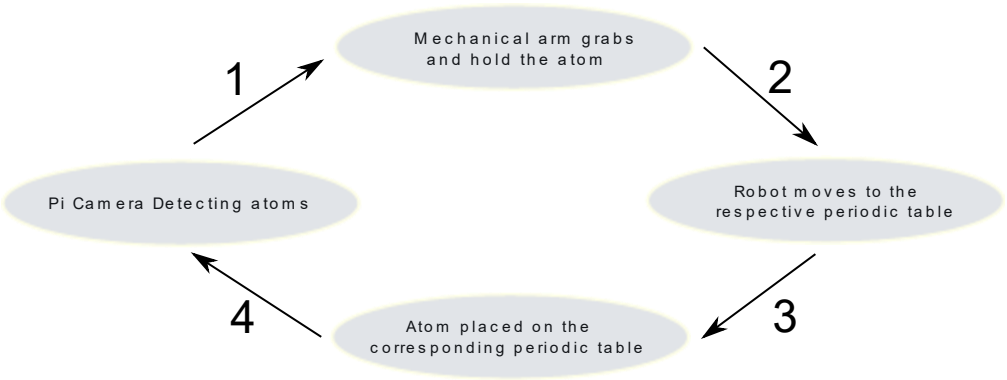



Figure 4-1 General strategy

From the Figure 4-1, one can see there are four steps involved in classifying the atoms.

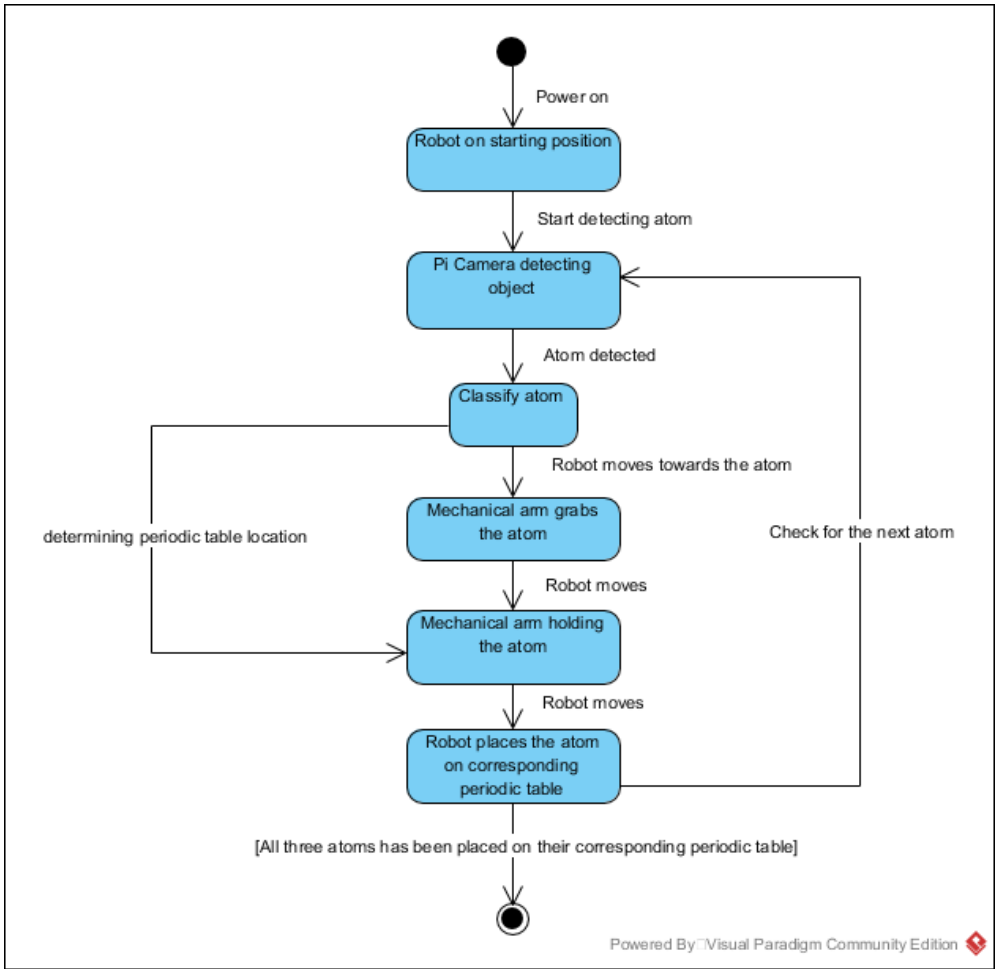



Figure 4-2 State diagram

The Figure 4-2 shows the state diagram. It reveals the starting and ending point, as well as the intermediate states. The state diagram starts when the robot's power switch in on.

I. The first state tells the robot is on the starting position.
II. The second state reveals that the Pi camera attached to the robot starts detecting atom.
III. The third state classifies the atom using object detection classifier and Pi camera.
IV. In the fourth state, the mechanical arm grabs the atom.
V. In the fifth state, the mechanical arm holds the atom. Here one can see, a link between state three and state five. The state three determines the periodic table location and sends the feedback information to state five.
VI. The state six follows by placing the atom on the corresponding periodic table. After placing the atom in state six, state changes from six to two. That means Pi camera on the robot is starting to detect the next object again.
VII. There is a ground condition set before the end position of the state diagram. The state diagram exits after placing all the three atoms on their corresponding periodic table.

### 4.1.1 Running the DC motors with the L298n motor drive

DC motors are responsible for the robot movement. To control the speed of the DC motor, we have used the L298n motor drive. We have developed a simple code to control both DC motors. The following sample code reveals how one can get started with DC motors and control the speed of the DC motors using the L298n motor drive.

```
// Motor A pin initialization on Arduino Uno board
```

```
int enA = 6; //enable pin of motor A to use PWM through motor drive, valued 0 to 255
int in1 = 7; //enable pins of motor A
int in2 = 8;

// Motor B pin initialization on Arduino Uno board

int in3 = 9; //enable pins of motor B
int in4 = 10;
int enB = 11; //enable pin of motor B to use PWM through motor drive, valued 0 to 255

void setup()

{

  // Set all the motor control pins to outputs
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);

  Serial.begin(9600);

}

//=========================================================
void forward()

{
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  analogWrite(enA, 215);

  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  analogWrite(enB, 215);

}

void right() {
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  analogWrite(enA, 215);

  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  analogWrite(enB, 215);

}

void left() {

  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  analogWrite(enA, 215);

  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
```

```
  analogWrite(enB, 215);

}

void stop() {

  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  analogWrite(enA, 0);

  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  analogWrite(enB, 0);

}

void backward() {
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  analogWrite(enA, 215);

  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  analogWrite(enB, 215);

}

void loop()

{

  forward();
  //backward();
  //left();
  //right;
  //stop();

}
```

Sample code to run the DC motors using the L298n motor drive

At first, we have initialized all the pins that we are using over the Arduino Uno board. Then at setup, we have defined the type of pins, whether input or output. At the loop function, we are calling the forward() function in the code. Another function has been commented to test the forward() function only. Then after testing the forward() function, we have commented on the forward() function and uncommented the backward() function to test. Thus, one by one, we have tested our functions.

### 4.1.2 Running the servo motors of the robot arm

The mechanical arm of the robot consists of four servo motors. First, we needed to learn and find a way to control the speed of the servo motors. Unlike DC motors, servo motors have built in mechanics to control motor speed. Here, we have developed a simple code that controls the servo motors.

```
// including Servo library file to the code
#include <Servo.h>

// Declaring servo pins
int servoPin1 = 2;
```

```
int servoPin2 = 3;
int servoPin3 = 4;
int servoPin4 = 5;

// Creating servo objects
Servo Servo1;
Servo Servo2;
Servo Servo3;
Servo Servo4;

int i; // initialization of a variable for loop control

void setup() {
  // Attach the servos according to the pin number
  Servo1.attach(servoPin1);
  Servo2.attach(servoPin2);
  Servo3.attach(servoPin3);
  Servo4.attach(servoPin4);

}
void loop() {

  servo1();
  //servo2();
  //servo3();
  // servo4();

}

void servo1() {
  for (i = 10; i < 150; i += 1) {
    Servo1.write(i);
    delay(15);

  }
  for (i = 150; i > 10; i -= 1) {
    Servo1.write(i);
    delay(15);

  }
}

void servo2()

{
  for (i = 0; i < 30; i += 1) {
    Servo2.write(i);
    delay(15);

  }

}

void servo3()

{
  Servo3.write(90);

}
```

```
void servo4() {
  for (i = 20; i < 120; i += 1) {
    Servo4.write(i);
    delay(5);
  }
}
```

Sample code to run the servo motors

To work with the servo in Arduino IDE, we need to include the workable library. In this case, we have added the Servo.h header file at the beginning of the code. Then, servo pins are declared. After that, we have created the servo objects, and then we have assigned the previously declared servo pins to the servo objects. In the loop, we can see there are four functions to be called. Each function corresponds to one of the four servos.

By calling the function one at a time as written in the code, we can check whether the particular servo of the robot arm is functioning or not. Putting delays in the loop enables us to maintain the speed of the servo motor.

### 4.1.3 Object detection using Pi Camera

To distinguish between the atoms, whether it is Redium, Greenium, or Blueium, we need to find a way to detect the object. For that, we are using the Raspberry Pi Camera module. As mentioned earlier, we are the Python programming language. It is used to control the Pi Camera module. Now we can follow a simple Pi camera code that captures an image. Simple Pi camera code in python:

```
From picamera import PiCamera
camera =  PiCamera()
camera.resolution = (800,600)
camera.start_preview()
camera.capture('picam.jpg')
```

Simple PiCamera code in Python to capture an image

First, we need to import the PiCamera library to the code. Then we need to declare the type of camera we are using which is PiCamera(). After that, we have set camera resolution 800x600. Point to be noted that, the Pi camera supports up to 1920x1080. The function camera.start_preview() start the Pi camera preview, where the camera.capture() function capture the image. Here, we can notice that we have named the image file as picam and the image format is jpg.

### 4.1.4 Communication between Arduino and Pi

As mentioned earlier, the robot uses the Arduino Uno board to control the DC motors on the wheels and servo motors on the mechanical arm. The Raspberry Pi does object detection using the Pi camera. We have established an I2C bus communication between the Arduino Uno and Raspberry Pi. The following Table 4-1 shows the pin connections between them.

| Raspberry Pi 3 Model B v1.2 | Arduino Uno |
|---|---|
| GPIO2 (SDA1) | Analog A4 |
| GPIO3(SCL1) | Analog A5 |
| GPIO6(Ground) | Ground |

Table 4-1 Pin connection table for $I^2C$ communication

In this case, we have connected the Pi's GPIO2 (SDA1) pin to the Analog A4 pin of Arduino Uno board, and Pi's GPIO3 (SCL1) pin to the Analog A5 pin of Arduino Uno board shown in the Figure 4-3.

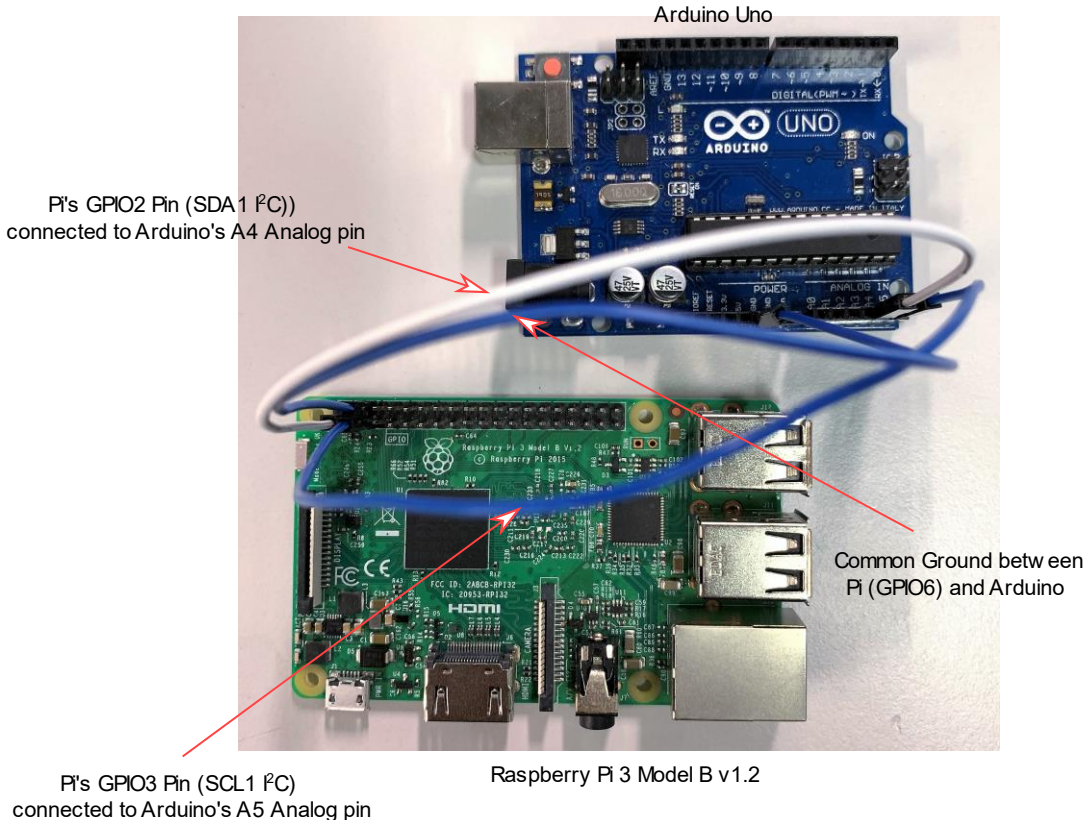



Figure 4-3 Raspberry Pi and Arduino using I²C Bus Connection

It is essential to make a common ground between them to fulfil the communication requirement.

# 4.2. Setting up Raspberry Pi and Pi camera

## 4.2.1 Setup and Installation of Debian OS

We already know that; Raspberry Pi operates in Debian OS. In our case, we have used the latest version of the Debian OS suitable for Raspberry Pi, which is Stretch with desktop and other recommended software.

One can easily download the Stretch OS with desktop from the Raspberry Pi website from the internet. After downloading the file, we need to unzip the file. The unzipped file contains the .iso image of the OS.

## 4.2.2 Setup Raspberry Pi

After successfully installing the Pi in the memory card, it is time to connect to the Raspberry Pi with a Laptop or external monitor. In our case, we have used both. So, now we will discuss the steps involved to do that.

In the case of using an external monitor and Pi desktop version OS, it is easy to configure. We need to connect the monitor's HDMI cable to Pi's HDMI port.

To connect the Pi with a laptop or other computer, we need to enable ssh connection in Pi. There are two ways to do that. One is creating a text file named ssh without any extension mentioned inside the memory card of Pi. Another one is enabling ssh from Pi.

There are two ways to enable the ssh on Pi depending on the type of OS. If the Pi is headless, then we have to write command in the terminal. Headless Pi means that the OS of the Pi has no Desktop appearance like Windows 10. Sometimes, we have used the Pi using an external monitor, and sometimes we have connected the Pi through an ethernet cable to the laptop. So we will cover both of the steps.

First, we will discuss the desktop version Pi attached to a monitor through the HDMI port. To enable ssh, we have to open a terminal window and write command raspi-config and hit the enter button of the keyboard shown in Figure 4-4.

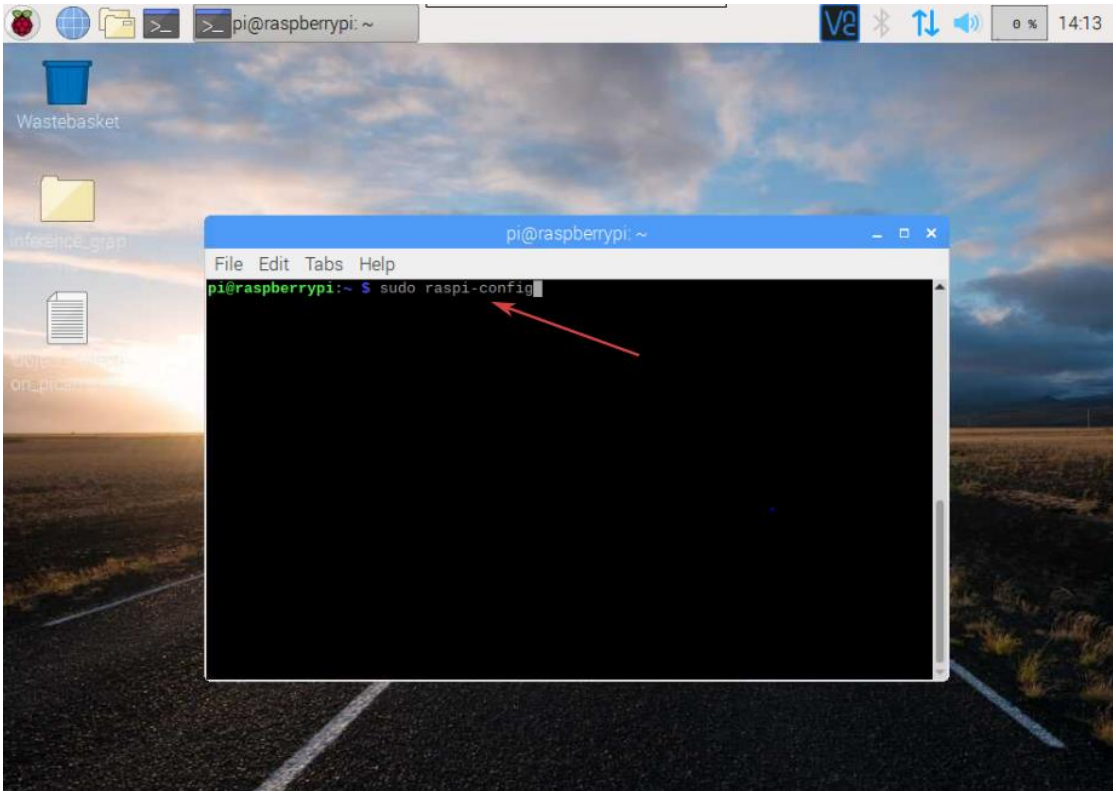



Figure 4-4 Configuring Raspberry Pi through writing raspi-config command in the terminal

The mentioned command will go to the configuration menu of Raspberry Pi shown in Figure 4-5. From there, one has to go to the ssh option from the configuration menu and enable it shown in Figure 4-6.

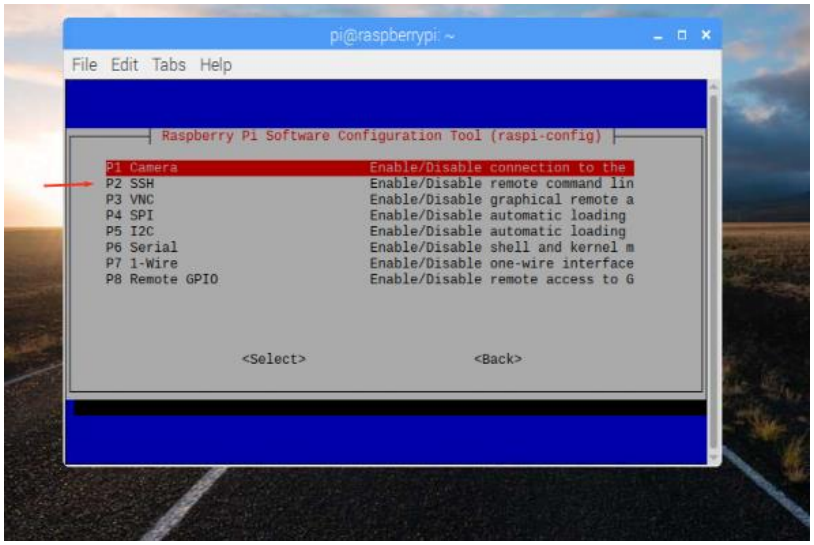



Figure 4-5 Enabling ssh on Pi method 1(1)

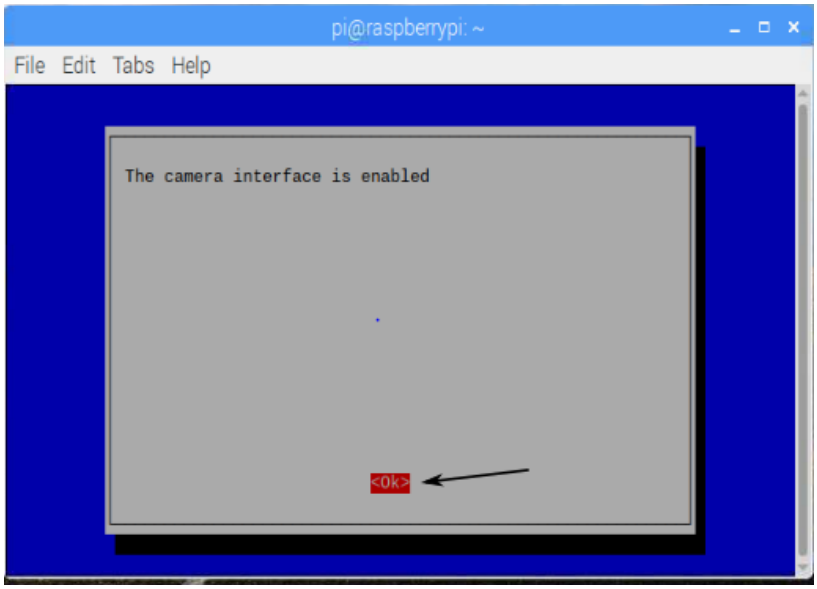



Figure 4-6 Enabling ssh on Pi method 1 (2)

There is another way to enable ssh. To do that we have to click on the leftmost Pi button. From there, we have to go down to the preference menu, and from there, we have to click upon the Raspberry Pi Configuration shown in Figure 4-7.

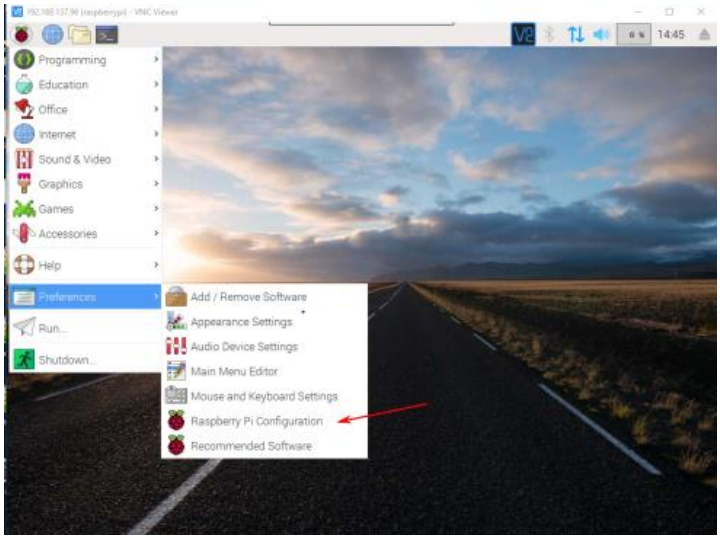


Figure 4-7 Enabling ssh on Pi method 2(1)

After clicking upon the Configuration, a new window will appear. From there, we have to go to the Interfaces menu. After putting the selection menu upon the Enabled button, we have to click OK, and the SSH for Pi is ready to use shown in Figure 4-8. A system reboot is required here.

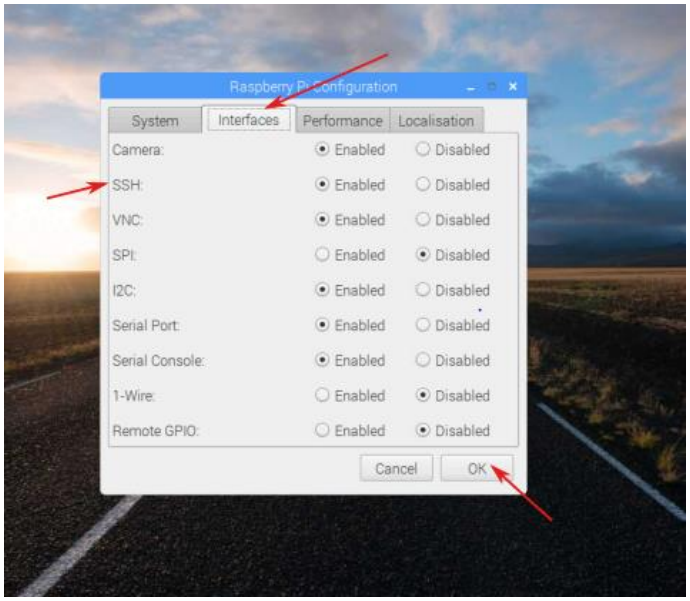


Figure 4-8 Enabling ssh on Pi method 2(2)

### 4.2.3 Connecting Pi to Laptop through ssh and VNC viewer

In our thesis, we have worked on the laptop most of the time. We are connecting the Pi to our laptop through ethernet LAN port. As the laptop we are using does not have built-in Ethernet LAN connectivity, so we have used ethernet to USB converter.

As mentioned earlier, we have already enabled ssh on Raspberry Pi. Now, to connect to the Raspberry Pi by ssh command first we need to know the IP address of the Raspberry Pi. In the case of using an external monitor, it is straightforward. We have to go to the terminal menu and type ifconfig.

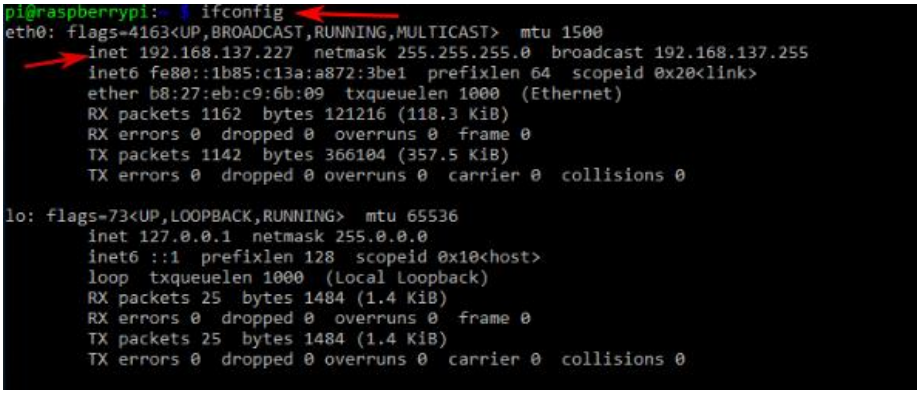



Figure 4-9 Connecting to Pi through ssh (1)

The result of the command will return the inet address, which is the IP address of the Pi shown in Figure 4-9. However, in most of the cases, we had to connect the Pi without any external monitor. In those cases, we do not know the IP address of the Pi from the beginning. We are going to discuss this issue now.

To know the IP address of the Raspberry Pi from Windows 10, we need to go to the command menu of Windows 10. By typing cmd in the search box, the command prompt will appear. Then we need to enter into the command prompt and issue the following command also shown in Figure 4-10.

```
ssh pi@raspberrypi.local
```

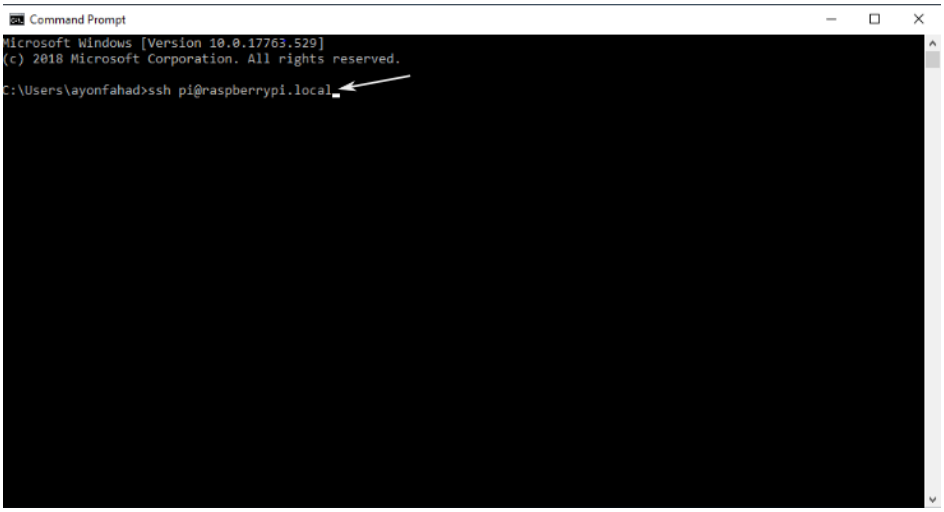



Figure 4-10 Connecting to Pi through ssh (2)

The issued command will automatically search for the raspberry pi connected to the laptop and ssh into it. After providing the password, we can log into Raspberry Pi from the command prompt of Windows 10 shown in Figure 4-11. Then we have to enter the password. After that, as mentioned earlier, we have to issue ifconfig in the command prompt.

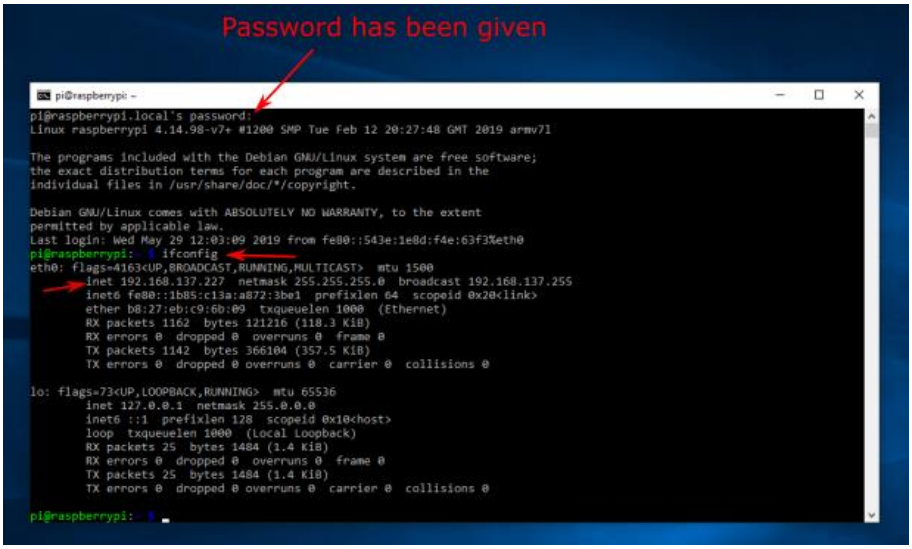



Figure 4-11 Connecting to Pi through ssh (3)

Here, the inet address is the IP address of the Raspberry Pi. VNC Viewer enables us to connect the Raspberry Pi through IP address and let us operate the Pi and the system OS (Windows 10, MacOS) in a parallel fashion. In our case, we have used an ethernet cable connected to the USB port of the laptop to establish a connection.

In most of the cases, we have faced problems connecting the Pi to the laptop through the ethernet cable. Because sometimes the IP address which is provided first by the ifconfig command does not always work in one chance. We need to follow a few steps to get the IP address of the Pi accurately. That IP address will be used to connect the Pi to Windows or MacOS through the VNC Viewer. First, we have to go to the Networks and settings options in Windows 10 and click upon it shown in Figure 4-12.

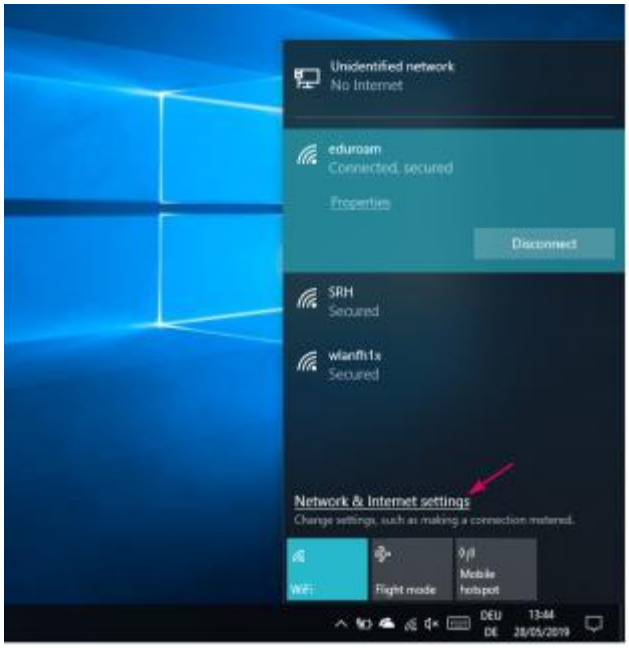



Figure 4-12 Troubleshooting Ethernet to Pi through ssh (1)

By clicking on the Network & Internet settings option, it will take us to the Status menu. From there, we have to click on the Change adapter options shown in Figure 4-13.

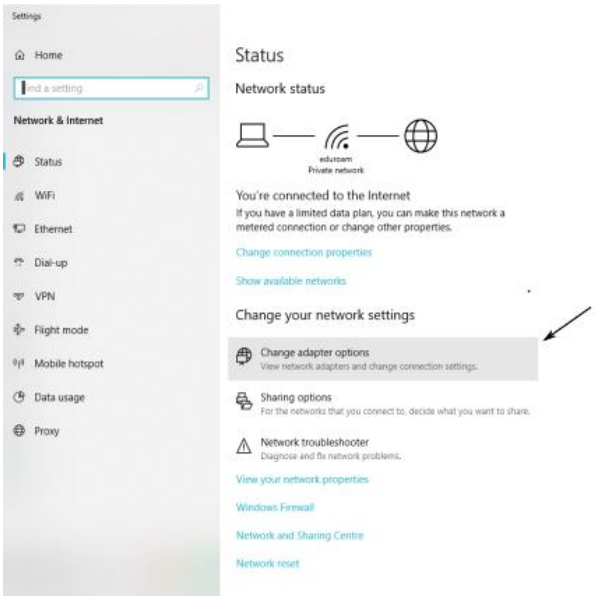



Figure 4-13 Troubleshooting Ethernet to Pi through ssh (2)

Then we have to go to the Properties of our wifi connection. In the Figure 4-14, we already see the Ethernet is connected. However, it is not getting the Pi's IP address from the Internet.

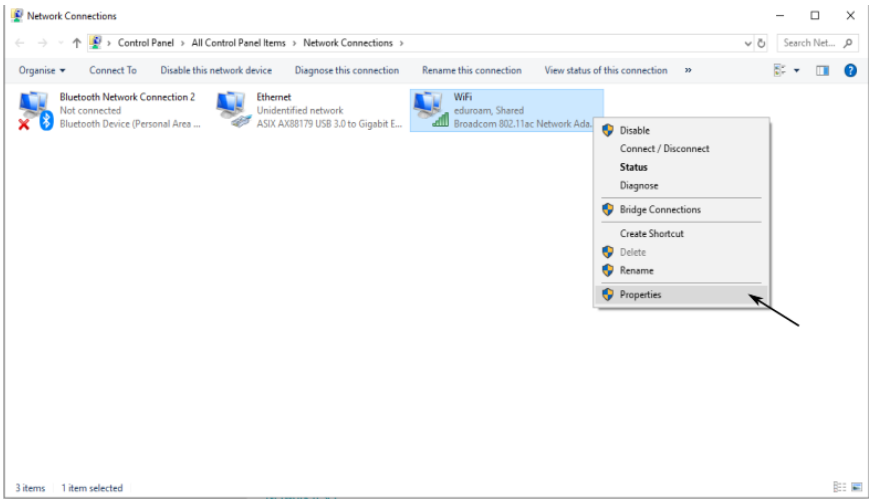



Figure 4-14 Troubleshooting Ethernet to Pi through ssh (3)

After that, we have to go to the Sharing menu. There we have to un-check in the Internet Connection Sharing option shown in Figure 4-15.

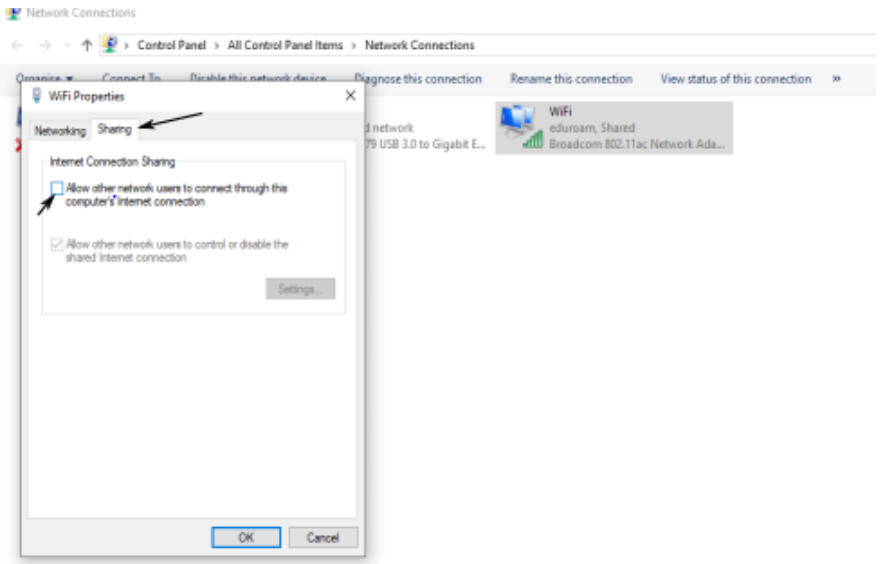



Figure 4-15 Troubleshooting Ethernet to Pi through ssh (4)

After un-checking the Internet Connection Sharing, we need to disconnect the ethernet cable connected to the USB and connect again. Now we are going to the Properties option of Wifi again, as shown earlier. This time we will check the Internet Connection Sharing option and click OK shown in Figure 4-16.

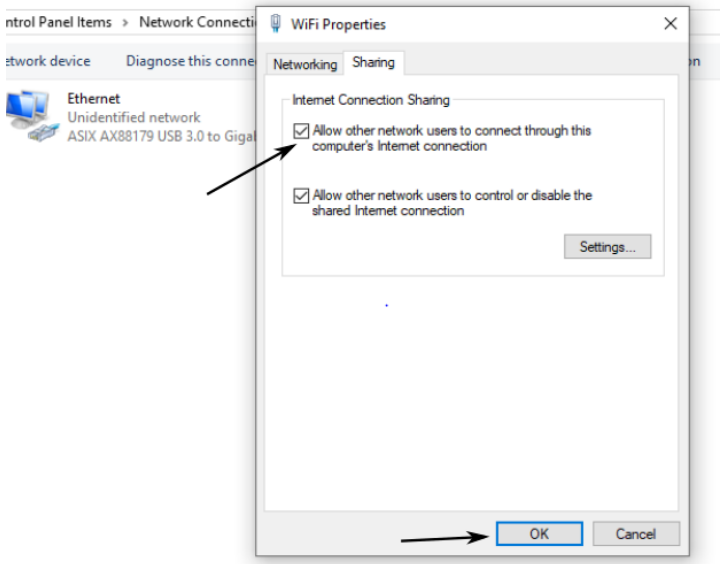



Figure 4-16 Troubleshooting Ethernet to Pi through ssh (5)

Now we will get the correct IP address of Raspberry Pi which we are going to use in VNC Viewer to connect to the Raspberry Pi. We have to follow the same steps, which is writing ifconfig in the terminal of the Pi. After putting the inet address in the VNC Viewer, we can get into the Pi.

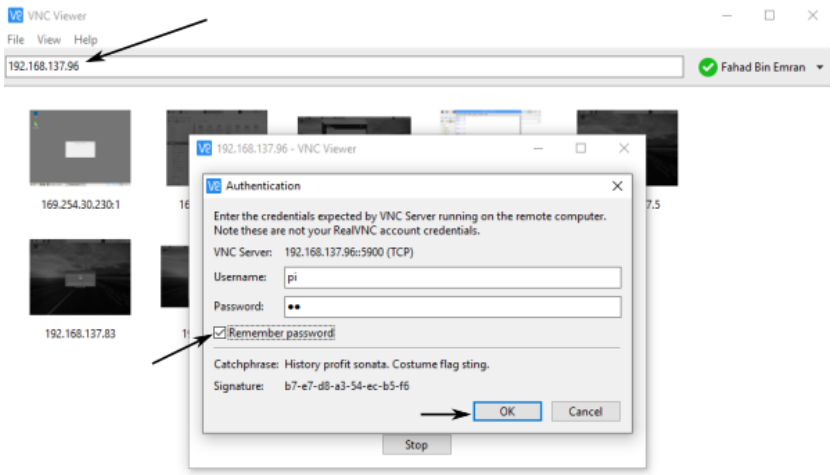



Figure 4-17 Logging into Pi through VNC Viewer (1)

We need to put the username and password of our Pi shown in Figure 4-17. Remember the password option will only be useful if we are working on the same WiFi. In that case, the inet address for the Pi will be the same, and we do not have to put the same username and password again.

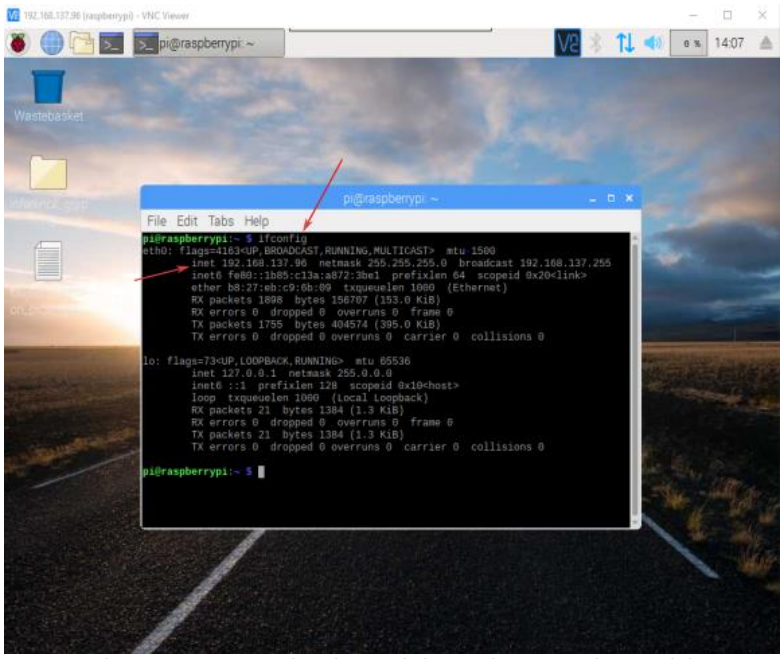



Figure 4-18 Logging into Pi through VNC Viewer (2)

After putting the username and password, we can log into the Pi's desktop window in VNC Viewer shown in Figure 4-18. Here, we are verifying the inet address again by writing ifconfig on the terminal window.

### 4.2.4 Installation of Raspberry Pi Camera

The installation of the Raspberry Pi Camera is pretty simple and straightforward. First, we have to establish the physical connection (shown in Figure 4-19) by merely attaching the Camera flex cable to the Raspberry Pi board's camera pin.

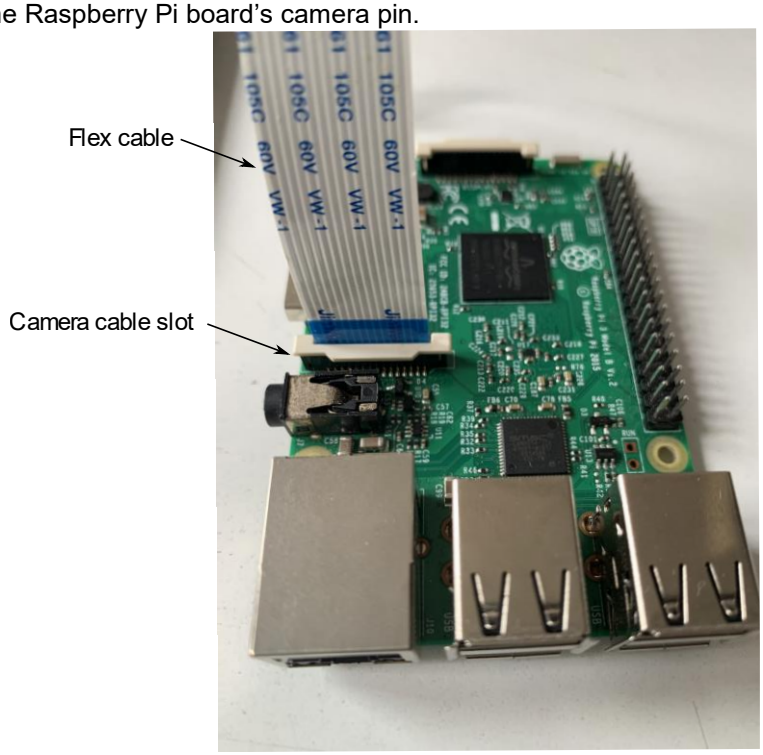



Figure 4-19 Connecting Pi camera's flex cable on Pi's camera slot

Then just like enabling ssh, it is essential to enable the Camera from the Raspberry Pi Configuration shown in Figure 4-20.

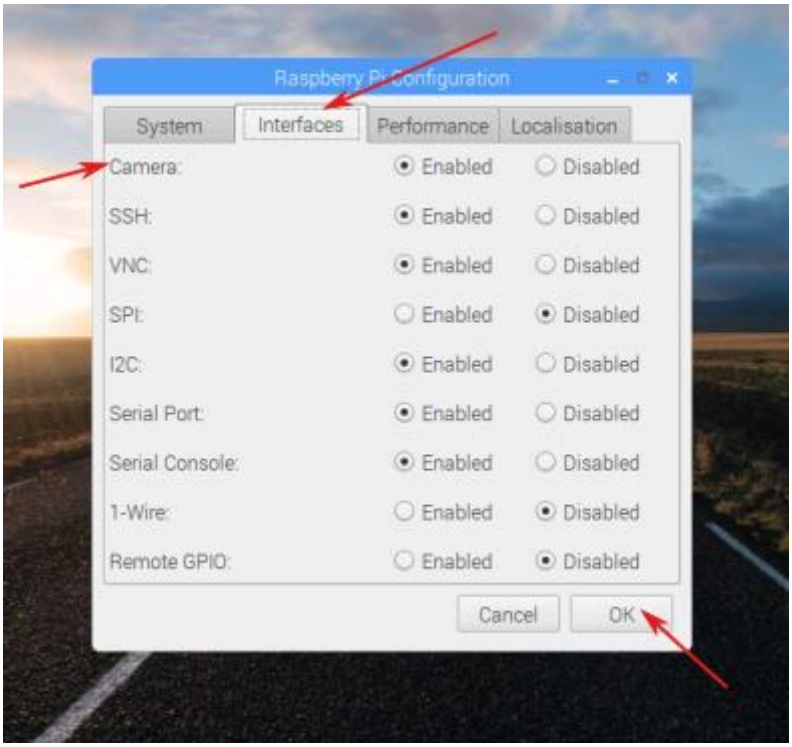


Figure 4-20 Enabling Pi Camera from Pi Configuration

After enabling the camera, it is required to make a reboot. After rebooting the system, we need to issue two commands in the terminal to update and upgrade the Pi so that, the camera can use all the latest libraries and functionalities. To update and upgrade the Pi, we need to issue the following commands.

```
sudo apt-get update
sudo apt-get upgrade
```

## 4.3. Installation of TensorFlow in Raspberry Pi

As we are going to use the Pi Camera, so it is evident for us to go for the desktop version of the OS. Otherwise, in headless Pi configuration, if we connect the Pi with a laptop through an ethernet connection, the display window of the camera does not show up. So, it is essential to use an HDMI supported monitor. There are a few essential steps to follow in order to install TensorFlow in Pi. They are as follows:

1. Updating the Pi
2. Installing TensorFlow
3. Installing OpenCV
4. Compiling and installing Protobuf
5. Setting up TensorFlow directory structure and PYTHONPATH variable
6. Creating a label map and configuring training data
7. Exporting Inference Graph

Considering a Raspbian OS has been already installed in Pi; first, we need to go to the terminal window and start writing some commands.

### 4.3.1 Updating Pi

As mentioned before, before doing anything, it is a wise decision to install the latest updates and upgrades for the Pi. By issuing following commands in the terminal window

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

Moreover, while installing packages on Pi if an error occurs the by default step to do is run the sudo apt-get update command. It works most of the time as all the necessary packages on the Pi get the current updates.

### 4.3.2 Installing TensorFlow

Now, we need to create a folder in which we will eventually install the TensorFlow. Before that, it is crucial to get track of the working directory. So first we need to go to the standard directory which is

```
cd /home/pi
```

After getting inside the pi directory, we are creating a folder named as tf.

```
mkdir tf
cd tf
```

As we already are inside the tf folder. It is time to get the installation file of TensorFlow. We are using a pre-built wheel file compatible with the Pi we are using. To download the installation file in the terminal window, we write the following command

```
wget https://github.com/lhelontra/tensorflow-on-
arm/releases/download/v1.8.0/tensorflow-1.8.0-cp35-none-
linux_armv7l.whl
```

Considering the file has been downloaded, it is time to install the file in Pi. To install the TensorFlow file, we write the following command

```
sudo pip3 install tensorflow-1.8.0-cp35-none-linux_armv7l.whl
```

To work on the TensorFlow couple of external packages and support packages are needed. These packages help the TensorFlow API to do its job. It is better to install them one at a time. By this, it will be easier to understand if any error occurs during the installation.

```
sudo apt-get install libatlas-base-dev
sudo pip3 install pillow
sudo pip3 install lxml
sudo pip3 install jupyter
sudo pip3 install matplotlib
sudo pip3 install cython
sudo apt-get install python-tk
```

### 4.3.3 Installing OpenCV

As we already know, TensorFlow supports a couple of deep learning framework. OpenCV is one of them. It is considered to be less vulnerable to errors. Many computer vision enthusiasts widely use it because of its robustness and easiness. Similar to TensorFlow, to install the OpenCV, we need to install some external packages as well as support packages.

```
sudo apt-get install libjpeg-dev
sudo apt-get install libtiff5-dev
sudo apt-get install libjasper-dev
sudo apt-get install libpng12-dev
sudo apt-get install libavcodec-dev
```

```
sudo apt-get install libavformat-dev
sudo apt-get install libswscale-dev
sudo apt-get install libv4l-dev
sudo apt-get install libxvidcore-dev
sudo apt-get install libx264-dev
sudo apt-get install qt4-dev-tools
```

After installing all the above packages successfully now, we can issue the following command to install OpenCV

```
pip3 install opencv-python
```

### 4.3.4 Compiling and installing Protobuf

Protobuf, in other words, Proto Buffer is kind of a technique that opens the door for serializing structured data. It is mainly beneficial to use in case of programs communicating with each other for doing particular actions such as storing data. Google uses it in TensorFlow software. Protobuf has certain advanced features. They are listed below

1. Language neutral

   It is language neutral as the programmer can define the structure of the data and necessary services. In simple words, the programmer writes the code containing the structure and services in a Proto definition file named .proto and start compiling with protoc.

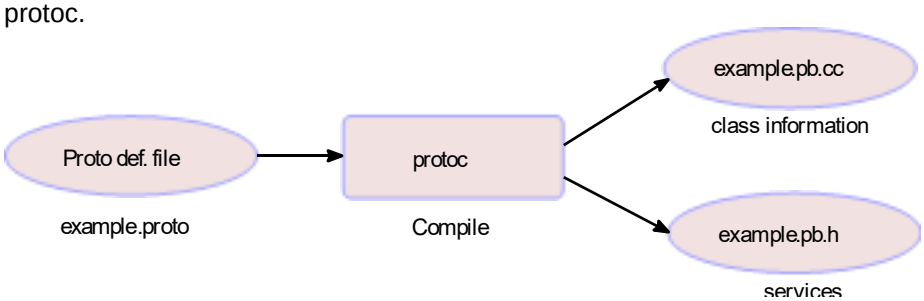



Figure 4-21 Compiling Proto def. file

   After the compilation, it creates two additional files. C++ file contains the class information, and the header file contains the services shown in Figure 4-21.

2. Platform neutral

   There are two versions of Protobuf. Proto2 supports the code generator facility for Objective C, C++, Python, and Java shown in Figure 4-22. Where the updated version Proto3 supports GO, Ruby, Dart, C#.

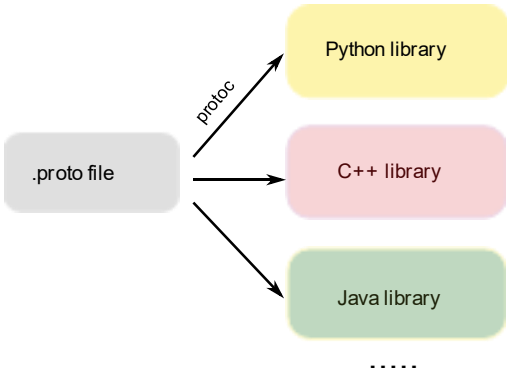



Figure 4-22 Proto file supported platforms

To compile the Protobuf first, we need to fetch the packages from the web. In the terminal window to Pi command

```
sudo apt-get install autoconf automake libtool curl
```

Now we need to download the Protobuf. It is advised to use the most recent version of Protobuf. In our case, we have downloaded Protobuf v3.5.1 from the GitHub repository. Worth to mention, we need to keep track of the directory of the Pi. For instance, now we are in /home/pi/tf directory. By issuing following command

```
wget
https://github.com/google/protobuf/releases/download/v3.5.1/protob
uf-all-3.5.1.tar.gz
```

The downloaded file is in tar format which has to unpacked for doing further process. Now we will unpack it and change our current directory into it by issuing the commands.

```
tar -zxvf protobuf-all-3.5.1.tar.gz
cd protobuf-3.5.1
```

After getting inside the unpacked Protobuf-3.5.1 directory, there are several steps to do. First, we need to configure the build. Then, we need to issue make command to issue the build process on the configured package.

```
./configure
make
```

After the build process completes, it is essential to check whether there has been an error that occurred in the make process. It is a time-consuming process that can take more than 100 minutes.

```
make check
```

The process might exit out with errors. In that case, we can retry the command a couple of times, and if there is still no success, we can skip the check step. It is time, for now, to install the Protobuf by issuing the following command

```
sudo make install
```

After the installation has completed, we need to change our directory to the Python folder. Then the library path has been exported

```
cd python
export LD_LIBRARY_PATH=../src/.libs
```

Few more implementation and path commands needed to be executed before finish installing Protobuf. They are:

```
python3 setup.py build --cpp_implementation
python3 setup.py test --cpp_implementation
sudo python3 setup.py install --cpp_implementation
```

```
export PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=cpp
export PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION_VERSION=3
```

At last, loading the configuration file ends the compiling and installing process of Protobuf. After that, it is advised to restart the Pi. Commands are as follows

```
sudo ldconfig
sudo reboot now
```

### 4.3.5 Setting up TensorFlow directory structure and PYTHONPATH variable

As we already have finished installing all the necessary packages, now we are going to set up the TensorFlow directory for our program. Again, we have to write some necessary commands to fulfil our necessity.

```
mkdir tensorflow1
cd tensorflow1
```

The mkdir command will make a tensorflow1 folder on the /home/pi/ directory. Now we are going to download the TensorFlow repository from GitHub by issuing following command

```
git clone --recurse-submodules https://github.com/tensorflow/models.git
```

We need to set the PYTHONPATH variable that directs to some of the directories of the model repository that we have just downloaded from to make the TensorFlow run properly. These environment variables must be needed to set every time the terminal window restarts. So, it is beneficial to add some bash script to make the PYTHONPATH set correctly every time whenever there is a click on the terminal window. To do that we have to issue a command to open the bashrc file and add a couple of command line (shown in Figure 4-23) at the end of the file.

```
sudo nano ~/.bashrc
```

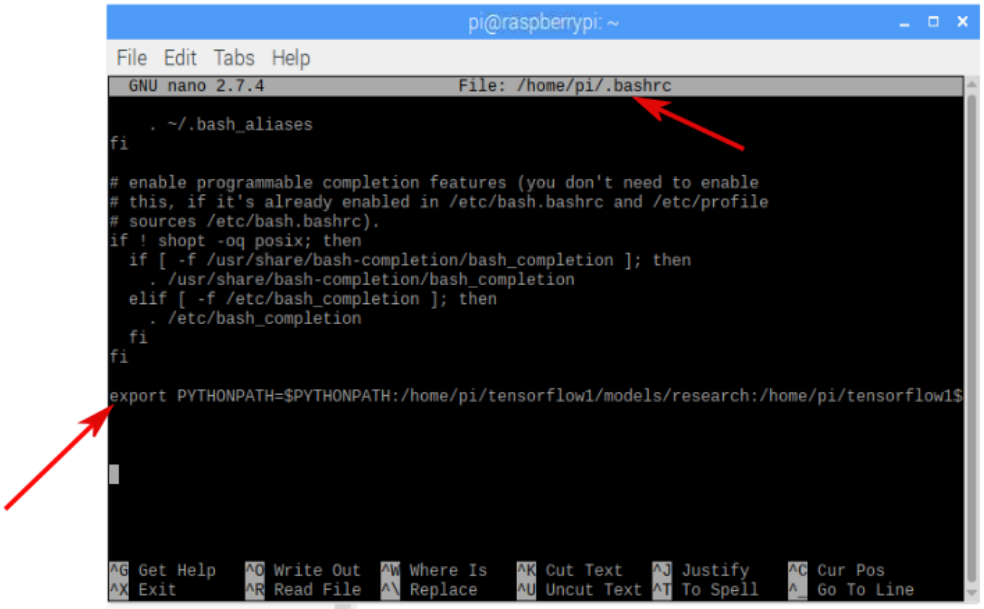



Figure 4-23 Setting up PYTHONPATH in .bashrc file

After saving the .bashrc file, we need to compile the Protobuf files which the object detection API will use. As mention earlier, using protoc, we can start compiling the .proto file. We need to keep in mind about our working directory this time. We have to run the Protoc command while we are in the research folder of the model respiratory.

```
cd /home/pi/tensorflow1/models/research
protoc object_detection/protos/*.proto --python_out=.
```

The mentioned command will create a filename_pb2.py file for each filename.proto file. Now we are going to change our directory to the object_detection folder in the research folder. By issuing

```
cd /home/pi/tensorflow1/models/research/object_detection
```

Now we are going to download one of the Google's pretrained models. Pretrained models have different speeds and accuracy depending on the model. Some models are suitable for high-performance hardware supported devices, and some are good for the devices having low processing power and hardware constraints.

As we are using Raspberry Pi, so we will go for a less hardware intensive model. We have to make a tread off between speed and performance here. In some models, we might get better speed but less accuracy and vice versa.

In our thesis, we have chosen the ssd_mobilenet model by coco, where coco is a large scale object detection, segmentation, and captioning dataset. In short features of coco are provided below so that we will get a clear idea about the dataset configuration.

1. Provides object segmentation.
2. Provides recognition in the context.
3. Supports superpixel stuff segmentation.
4. 330K images > (200K labeled).
5. 1.5 million object instances.
6. There are 80 object categories.
7. There are 91 stuff categories.
8. There are 91 stuff categories.
9. There are 91 stuff categories.

To download and unpack the ssdlite_mobilenet model by coco, we need to issue the following commands

```
wget
http://download.tensorflow.org/models/object_detection/ssdlite_mob
ilenet_v2_coco_2018_05_09.tar.gz
tar -xzvf ssdlite_mobilenet_v2_coco_2018_05_09.tar.gz
```

Now the model is unpacked in the object detection folder and ready to be used.

# 4.4. Setting up TensorFlow in Windows 10

## 4.4.1 Setup and Installation of TensorFlow

There are several steps involved to set up and install TensorFlow in Windows 10. We can install TensorFlow-CPU or TensorFlow-GPU. Training the object detection API's in TensorFlow-GPU is significantly faster than the other one.

In our case, we are using the TensorFlow-CPU version as the laptop we will be using has no TensorFlow-GPU support. The computer must meet two conditions to have to GPU support by TensorFlow.

1. The computer must have Nvidia GPU.
2. The Nvidia GPU supports TensorFlow.

The laptop we are using has an Intel Core i5 processor. There are some intel optimized steps to install the TensorFlow-CPU in the Intel processor. Intel has modified the TensorFlow framework using Intel Math Kernel Library for deep learning applications.

Anaconda covers the gap by providing scientific computing facilities integrated with Python. Anaconda can build the environment for TensorFlow in which Deep Neural Networks uses Intel Math Kernel Library. There are three versions available to install. One is from Anaconda Cloud, and another one is from the Intel channel. The last one is from Intel Distribution for Python. In our case, we have followed all the steps to make sure the TensorFlow-CPU is running correctly on our laptop.

In Anaconda prompt we need to write commands to install the TensorFlow-CPU with Intel optimization for maximum performance.

```
conda install tensorflow-mkl -c anaconda
conda install tensorflow -c intel
conda create -n IDP intelpython3_full -c intel
pip install intel-tensorflow
```

It might take a bit more space in our Laptop, but following this ensures that we have successfully installed TensorFlow from all channels. In some cases, already installed messages can appear on the terminal.

### 4.4.2 Train Object detection classifier

Before starting to train the object detection classifier, we have to follow a couple of steps. The steps are listed below.

1. Setting up object detection directory structure and Anaconda virtual environment.
2. Gathering and labelling pictures.
3. Generating training data.
4. Creating a label map and configuring training.
5. Training.
6. Export inference graph.

### 4.4.3 Setting up object detection directory and Anaconda virtual environment

We need to set up the object detection directory for TensorFlow Object Detection API. This API follows some particular structure of directories. Its GitHub repository contains all of those directories. It is essential to download several other python packages, and we also need to fix some PATH and PYTHONPATH variables.

At first, we need to create a folder in C: drive of the Windows 10. We can name the folder as tensorflow1. Now we need to download the TensorFlow repository from the following link.

```
https://github.com/tensorflow/models
```

After downloading the repository, we need to extract the file. The extracted file will create a models-master folder. Now we will rename the folder name from models-master to models. It will make the path directory less long in the future.

Now we are going to download a pre-trained classifier with a specific neural network. As mentioned earlier in Raspberry Pi, we are using ssd mobilenet by coco object detection

classifier, in this case, we will be using a kind of the same classifier named ssdlite_mobilenet_v2_coco_2018_05_09. First, we need to download the classifier from the following link.

```
http://download.tensorflow.org/models/object_detection/ssdlite_mobilenet_v2_coco_2018_05_09.tar.gz
```

Then extract the tar file. After extracting the tar file, we will put the extracted file inside the object detection folder, which is inside the research folder. Path of the extracted file will be like

```
C:\tensorflow1\models\research\object_detection
```

Now we need to download another object detection repository on which we will be working on from the following link.

```
https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10.git
```

After downloading and extracting the file, we need to copy the content of the file and paste it inside the object detection folder.

```
C:\tensorflow1\models\research\object_detection
```

We are going to train our object detection classifier to recognize Redium, Greenium, and Blueium atoms. After completing the steps mentioned above, we need to make some changes, like deleting some contents as they will be created again according to our needs. We need to delete

1. All files inside C:\tensorflow1\models\research\object_detection\images\test folder.
2. All files inside C:\tensorflow1\models\research\object_detection\images\train folder.
3. Both test_labels.csv and train_labels.csv files on the object_detection\images folder.
4. All files in C:\tensorflow1\models\research\object_detection\training folder.
5. All files in C:\tensorflow1\models\research\object_detection\inference_graph folder.

Considering Anaconda virtual environment has been installed in Windows 10 of the laptop we are using, now we are going to set up the Anaconda virtual environment for TensorFlow. It is needed to keep in mind that to modify and run commands, we need the administrator permission from the Windows 10.

So after starting the Anaconda prompt in Administrator mode, we need to write up following command.

```
conda create -n tensorflow1 pip python=3.5
```

This command will create a virtual environment named tensorflow1. Now we are activating the environment by issuing

```
activate tensorflow1
```

Now we need to install other necessary packages by issuing the following commands one by one.

```
conda install -c anaconda Protobuf
pip install pillow
pip install lxml
pip install Cython
pip install jupyter
pip install matplotlib
pip install pandas
pip install opencv-python
```

Just like in Raspberry Pi, we need to set the PYTHONPATH environment variable. We need to keep in mind that after activating the virtual environment tensorflow1, we need to set the PYTHONPATH. Accidentally if we get to exit from the virtual environment, we need to set the path again otherwise the thing will not work correctly. To set the PYTHONPATH, we need to write the following command.

```
set
PYTHONPATH=C:\tensorflow1\models;C:\tensorflow1\models\research;C:
\tensorflow1\models\research\slim
```

After setting the PYTHONPATH, we need to compile TensorFlow's Protobufs just like we did in case of Pi.

```
protoc                                                          --
python_out=. .\object_detection\protos\anchor_generator.proto .\ob
ject_detection\protos\argmax_matcher.proto .\object_detection\prot
os\bipartite_matcher.proto .\object_detection\protos\box_coder.pro
to .\object_detection\protos\box_predictor.proto .\object_detectio
n\protos\eval.proto .\object_detection\protos\faster_rcnn.proto .\
object_detection\protos\faster_rcnn_box_coder.proto .\object_detec
tion\protos\grid_anchor_generator.proto .\object_detection\protos\
hyperparams.proto .\object_detection\protos\image_resizer.proto .\
object_detection\protos\input_reader.proto .\object_detection\prot
os\losses.proto .\object_detection\protos\matcher.proto .\object_d
etection\protos\mean_stddev_box_coder.proto .\object_detection\pro
tos\model.proto .\object_detection\protos\optimizer.proto .\object
_detection\protos\pipeline.proto .\object_detection\protos\post_pr
ocessing.proto .\object_detection\protos\preprocessor.proto .\obje
ct_detection\protos\region_similarity_calculator.proto .\object_de
tection\protos\square_box_coder.proto .\object_detection\protos\ss
d.proto .\object_detection\protos\ssd_anchor_generator.proto .\obj
ect_detection\protos\string_int_label_map.proto .\object_detection
\protos\train.proto .\object_detection\protos\keypoint_box_coder.p
roto .\object_detection\protos\multiscale_anchor_generator.proto .
\object_detection\protos\graph_rewriter.proto .\object_detection\p
rotos\calibration.proto
```

We have encountered an error running the command. Because one of the proto files was not declared to convert into a pb2.py file. Then we added the file name at the last of the command following the format. Here on the command line, we have added calibration.proto command. The command written above will create the filename_pb2.py file for every filename.proto file in the following folder.

```
C:\tensorflow1\models\research\object_detection\protos
```

Now we need to change our working directory from object detection folder to the research folder by issuing the following command

```
C:\tensorflow1\models\research
```

From research directory, we will run the following commands on the Anaconda prompt

```
python setup.py build
python setup.py install
```

### 4.4.4 Gathering and labelling pictures

Now we are going to gather and label pictures to train our classifier. To train the object detection classifier by TensorFlow, we need hundreds of pictures. The images should have the objects we need to classify and other random objects. Images should be taken in various lighting conditions having a different background. We will edit every image one by one and label our desired object from the particular image.

In our case, we have taken in total 166 pictures of Radium, Greenium, and Blueium atoms which are in different background and lighting conditions. Then we have taken the whole 100% of the images and divided them into 20% test images and 80% train images. There should be

a variation of the images in both the test and train folder. That means images will have different backgrounds and lighting conditions. In our case, we put 26 test images in the test folder and 140 images in the train folder. For better understanding, link of the folder directory is below

```
C:\tensorflow1\models\research\object_detection\images\test
C:\tensorflow1\models\research\object_detection\images\train
```

After putting the test and train images on their respective folders now, it is time to label the images. With LabelImage software, we can easily label objects in an image. First, we need to select our working directory on the LabelImage software by clicking on Open Dir from the left side menu shown in Figure 4-24. It will be easier for us to label many images by selecting the working directory. We have to go through every image on the directory and label the objects by name, and we also have to draw a rectangular box over the object. To do that we will click upon Create/mRectBox button from the left menu and start placing the box upon the desired object to detect.

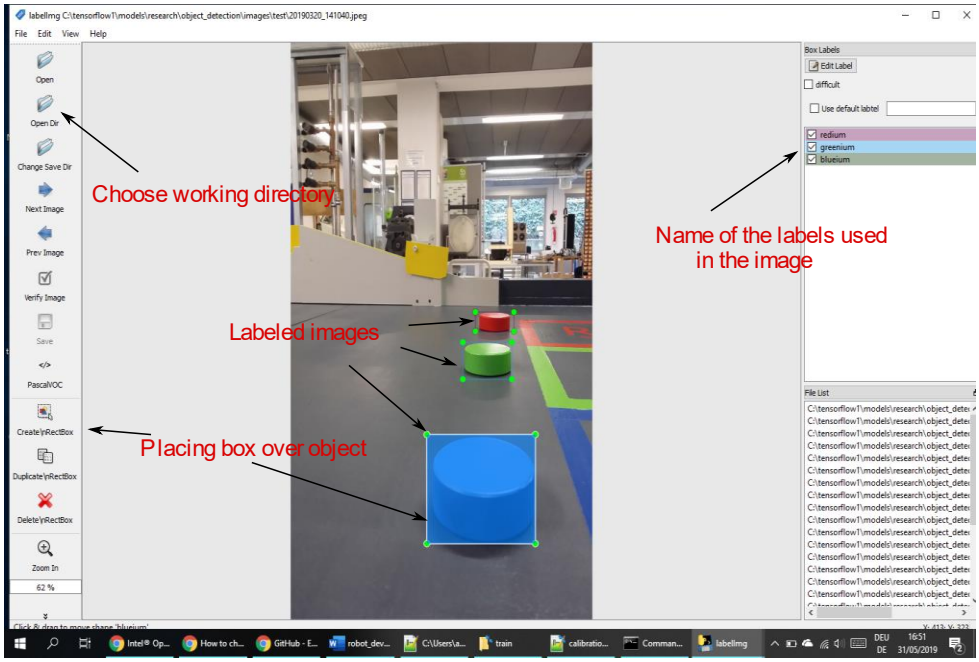



Figure 4-24 Labeling inamge using LabelImage

LabelImage also creates a .xml file for each image where the file contains label data. For the above image generated .xml file is given below.

```
<annotation>
        <folder>test</folder>
        <filename>20190320_141040.jpeg</filename>
        <path>C:\Users\ayonfahad\Desktop\test\20190320_141040.jpeg</path>
        <source>
                <database>Unknown</database>
        </source>
        <size>
                <width>720</width>
                <height>1280</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>redium</name>
                <pose>Unspecified</pose>
```

```
            <truncated>0</truncated>
            <difficult>0</difficult>
            <bndbox>
                <xmin>441</xmin>
                <ymin>598</ymin>
                <xmax>533</xmax>
                <ymax>641</ymax>
            </bndbox>
        </object>
        <object>
            <name>greenium</name>
            <pose>Unspecified</pose>
            <truncated>0</truncated>
            <difficult>0</difficult>
            <bndbox>
                <xmin>411</xmin>
                <ymin>665</ymin>
                <xmax>528</xmax>
                <ymax>747</ymax>
            </bndbox>
        </object>
        <object>
            <name>blueium</name>
            <pose>Unspecified</pose>
            <truncated>0</truncated>
            <difficult>0</difficult>
            <bndbox>
                <xmin>325</xmin>
                <ymin>870</ymin>
                <xmax>584</xmax>
                <ymax>1112</ymax>
            </bndbox>
        </object>
</annotation>
```

Generated .xml file overview

From the xml code, we can easily see that the software has generated xmin, xmax, ymin, and ymax values alongside with some details for each object of the image. We can check whether all the bounding box has been placed correctly or not from the object detection directory by issuing the command

```
python sizeChecker.py --move
```

The sizeChecker.py program already resides in the repository. In case there are errors regarding labelling the images, the sizeChecker.py code will place the images with a wrong bounding box in a folder inside test and train folders. Then we have to go through that folder with faulty labelled images and correct the bounding box and at last run the above command again until there is no error regarding bound boxes of the images.

Now we need to convert the xml files of the images into a csv file. To do that there is already a python program provided in the repository. For that, we can issue

```
python xml_to_csv.py
```

The mentioned command will create train_labels.csv and test_labels.csv file in the following directory

```
C:\tensorflow1\models\research\object_detection\images
```

Now we have to edit the generate_tfrecord.py file according to our requirements. File generate_tfrecord.py was already in the object detection repository. As we have three objects to detect which are Radium, Greenium, and Blueium, we will modify and add the following lines.

```
def class_text_to_int(row_label):
    if row_label == 'redium':
        return 1
    elif row_label == 'greenium':
        return 2
    elif row_label == 'blueium':
        return 3

    else:
        None
```

Editing generate_tfrecord.py

After modifying the generate_tfrecord.py file now, it is time to generate TFRecords. TFRecords acts as an input data for TensorFlow training model. We can do that by issuing the following commands

```
python generate_tfrecord.py --csv_input=images\train_labels.csv --image_dir=images\train --output_path=train.record
python generate_tfrecord.py --csv_input=images\test_labels.csv --image_dir=images\test --output_path=test.record
```

The commands will create test.record and train.record inside the following directory shown in Figure 4-25.

```
C:\tensorflow1\models\research\object_detection
```

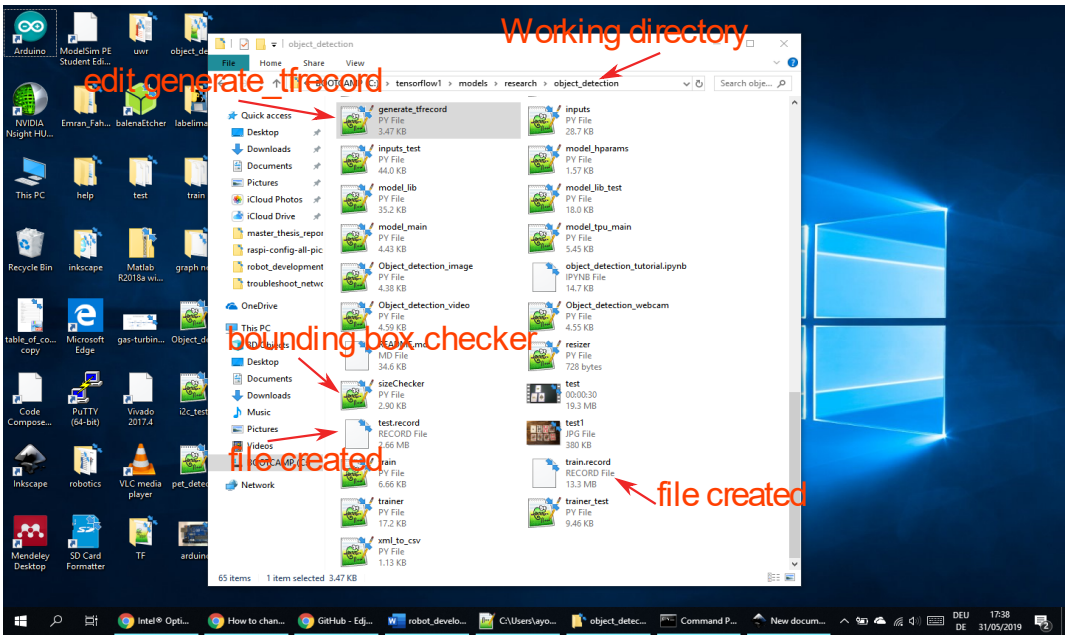



Figure 4-25 Object detection folder overview

## 4.4.5 Creating the label map and configuring training data

The label map helps to map the object according to the class name and ID number. It works as a helping hand for the trainer. In our case, we have created a label map by merely writing the following code in a text editor. We have to keep in mind that the label map serial of objects is as same as we have written in the generate_tfrecord.py file. For example, in the generate_tfrecord.py file, we have serialized the objects as Radium, Greenium, and Blueium. In the label map file, we will follow the same order.

```
item {
  id: 1
  name: 'redium'
}

item {
  id: 2
  name: 'greenium'
}

item {
  id: 3
  name: 'blueium'
}
```

Figure 4-26 Editing labelmap.pbtxt

After writing the code inside the file, we have named the file name as labelmap.pbtxt and put the file inside the following directory.

```
C:\tensorflow1\models\research\object_detection\training
```

Now it is time to configure the training model. As we are using the ssd_mobilenet_v2_coco model, so we will take the configuration file for that from the following repository folder.

```
C:\tensorflow1\models\research\object_detection\samples\configs
```

We will copy the ssd_mobilenet_v2_coco.config file and paste the config file in the following directory.

```
C:\tensorflow1\models\research\object_detection\training
```

After pasting the config file in the training folder, we need to make changes according to our needs. First, we need to change the number of classes shown in Figure 4-27. As our object detection classifier will detect three objects so in our case num_classes: 3

```
model {
  ssd {
    num_classes: 3
    box_coder {
      faster_rcnn_box_coder {
        y_scale: 10.0
        x_scale: 10.0
        height_scale: 5.0
        width_scale: 5.0
      }
    }
```

Figure 4-27 Changing the number of classes

Now we need to set the fine_tune_checkpoint path as shown in Figure 4-28.

```
fine_tune_checkpoint: "C:/tensorflow1/models/research/object_detection/ssdlite_mobilenet_v2_coco_2018_05_09/model.ckpt"
fine_tune_checkpoint_type:  "detection"
# Note: The below line limits the training process to 200K steps, which we
# empirically found to be sufficient enough to train the pets dataset. This
# effectively bypasses the learning rate schedule (the learning rate will
# never decay). Remove the below line to train indefinitely.
```

Figure 4-28 Finetuning checkpoint paths

We also need to modify the input_path and label_map_path shown in Figure 4-29.

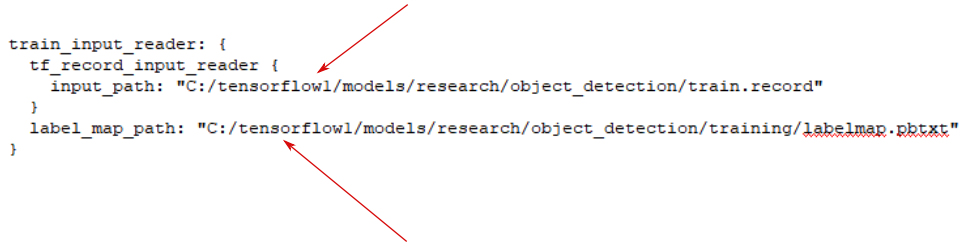



Figure 4-29 Modifying the input path and label map path for train input

Now it is time to define the number of text images. In our case, we are using 26 test images. So we need to change the number of test images according to our number shown in Figure 4-30.

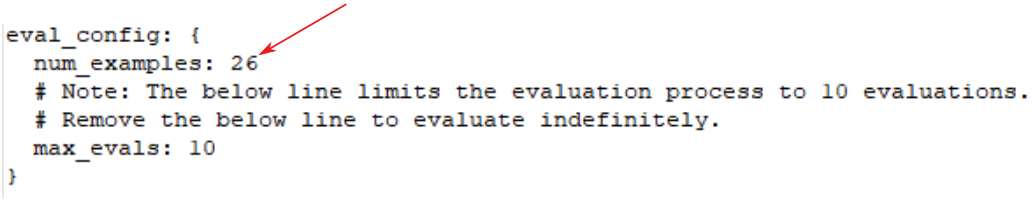



Figure 4-30 Modifying the number of test images

At last, we need to make some changes in the evaluation of the input reader section. It goes as shown in Figure 4-31

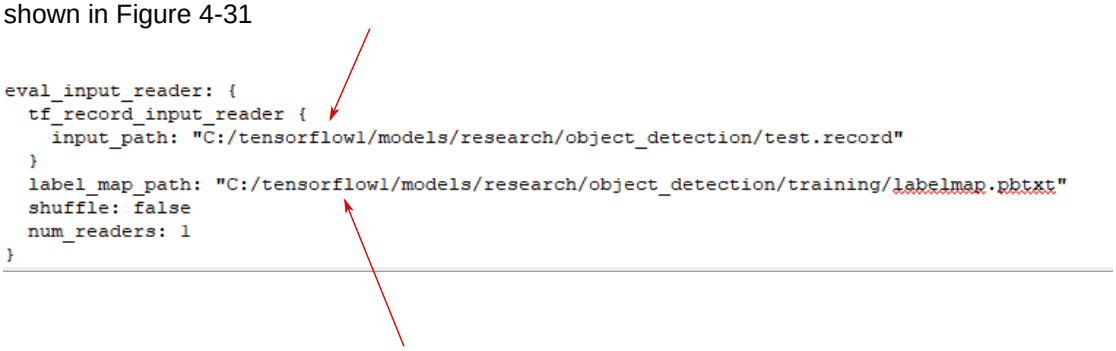



Figure 4-31 Modifying the input path and label map path for evaluation input

## 4.4.6 Running the Training

Finally, it is time to run the training. To start the training process, we need to issue the following command.

```
python train.py --logtostderr --train_dir=training/ --
pipeline_config_path=training/ssd_mobilenet_v2
_coco.config
```

The mentioned command will start the training process using TensorFlow. Almost 30 sec is needed to start the initialization. Then the real training begins. The global step for training the object detection classifier starts from step one. It took us around six days to run the training until the global step is 41189. TensorFlow automatically saves the record after a couple of iteration in the global step. Training reports has some information field. Especially the loss

indication. In the beginning, the loss is high, and as the steps count gets increased, the loss gets less eventually. The following Figure 4-32 shows a typical form of the terminal

```
INFO:tensorflow:global step 1: loss  =  4.0   (11.3 sec/step)
INFO:tensorflow:global step 2: loss  =  3.95 (14.3 sec/step)
INFO:tensorflow:global step 3: loss  =  3.99 (12.8 sec/step)
INFO:tensorflow:global step 4: loss  =  3.93 (10.3 sec/step)
INFO:tensorflow:global step 5: loss  =  3.96 (22.3 sec/step)
INFO:tensorflow:Recording summary at step 5.
```

Figure 4-32 Typical tensorflow training command window

Anytime we can start the training again as the recorded summary has all the details. So, in case we restart a training, the global step will start from the last recorded step. Moreover, we can exit from the terminal training windows at any time by pressing Ctrl+C, which creates an interrupt and quit the training process.

### 4.4.7 Exporting Inference Graph

After getting a satisfactory loss result (lower the better), we can quit the training and export the training into the inference graph. To generate an inference graph, we need to issue a command.

```
python export_inference_graph.py --input_type image_tensor --
pipeline_config_path training/ssd_mobilenet_v2_coco.config --
trained_checkpoint_prefix training/model.ckpt-41189 --
output_directory inference_graph
```

In the command, we can see that we have written the highest global step, which is 41189 generated in our training. To get the highest global step number, we need to go inside the following path.

```
C:\tensorflow1\models\research\object_detection\training
```

Then we have to look for the highest number of step generated on the folder shown in Figure 4-33.

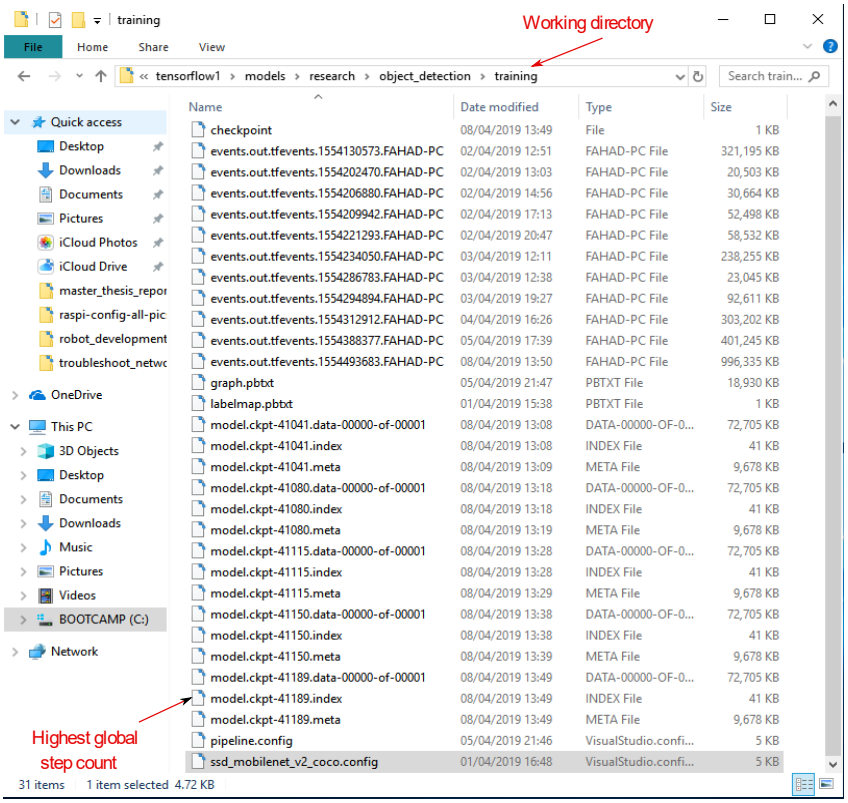



Figure 4-33 Finding the highest global count in training folder

The following folder contains the frozen inference graph.

```
C:\tensorflow1\models\research\object_detection\inference_graph
```

From the Figure 4-34, we can see the generated frozen inference graph containing the object detection classifier.

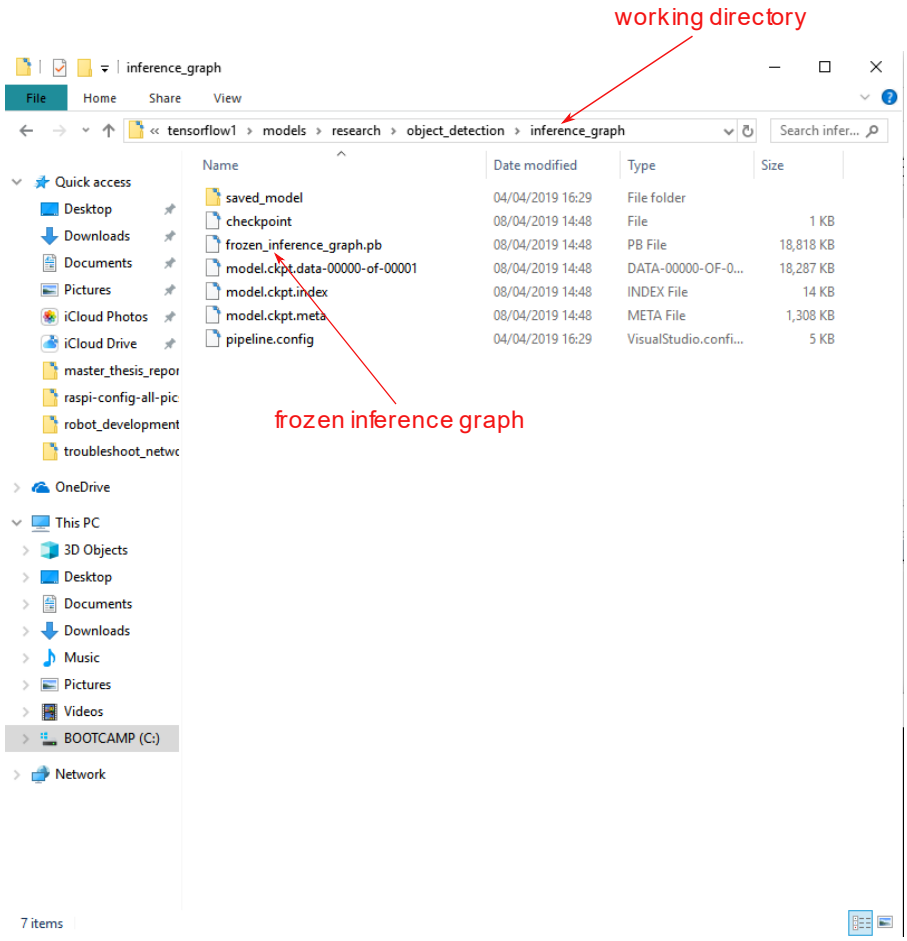



Figure 4-34 Generated frozen inference graph directory

# 4.5. Setting up Arduino Uno

## 4.5.1 Setup and Install Arduino IDE

As mentioned earlier, the Arduino IDE provides easy steps to write the code and upload it into the board. In our thesis, we have installed Arduino IDE in both Windows 10 and MacOS operating systems. There is also an online editor version available. One has to register an account and then can start using the online editor. In our case, we have downloaded the latest version of Arduino IDE from the following link

```
https://www.arduino.cc/en/Main/Software
```

The Arduino IDE supports Windows, MacOS, and Linux operating systems. After installing the IDE, the user can start writing the code inside a sketch window shown in Figure 4-35.

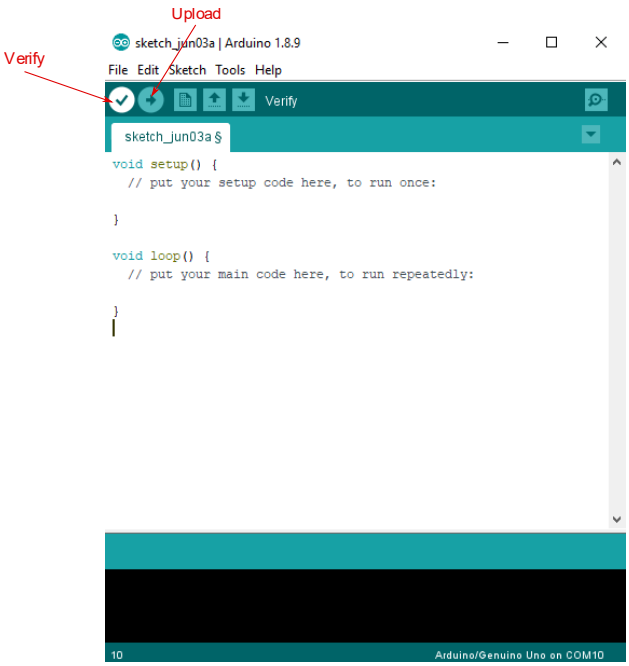


Figure 4-35 Typical Arduino IDE sketch window

After finished writing the code, the user can verify the code to check whether there is an error in the code. As shown in the Figure 4-35, the user has to click upon the verify button to verify the code. If there is no error in writing the code, the IDE will show done compiling. After compiling the code without any error, the user can upload the code into the Arduino board by clicking upon the Upload button provided in the Arduino IDE.

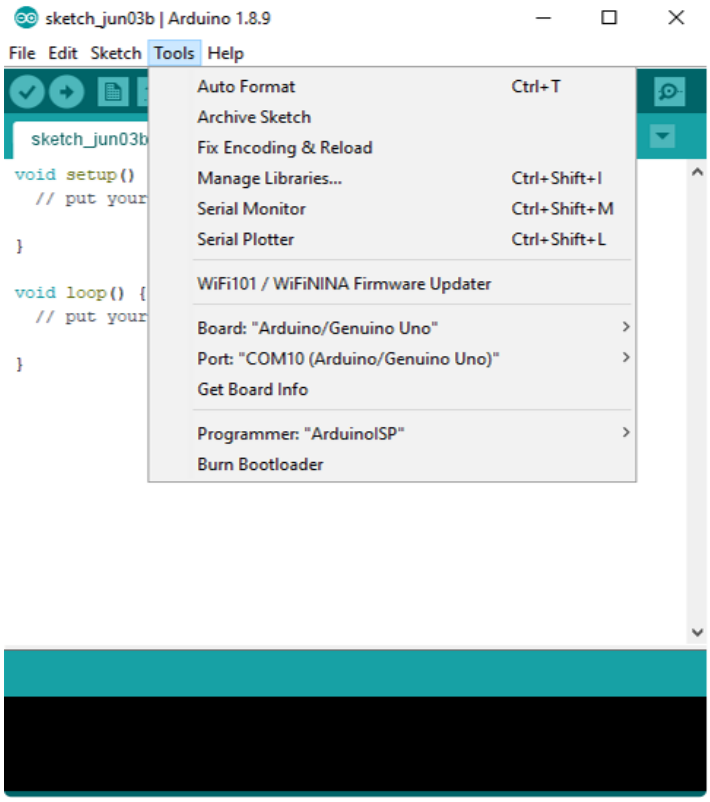


Figure 4-36 Setting up the Arduino IDE

Before uploading the code, we have to make sure that specific parameters are placed correctly like the Board and Port selection Tools menu shown in Figure 4-36. As we are using the Arduino

Uno board, so in Board option from Tools menu, we have selected Arduino/Genuino Uno. The port number should be assigned automatically whenever an Arduino board connects to a computer. However, sometimes, it has to be assigned manually. We have selected the Programmer as ArduinoISP.

## 4.6. Load Frozen graph in Raspberry Pi

As mentioned above, we have successfully generated a frozen dataflow graph for our object detection classifier. Now, we are going to put the graph inside the tensorflow1 directory of Raspberry Pi. We need to do a few additional steps as well.

First, we are copying the inference_graph folder from Windows 10's TensorFlow directory to Raspberry Pi's TensorFlow directory. We have renamed the folder atom factory shown in Figure 4-37.

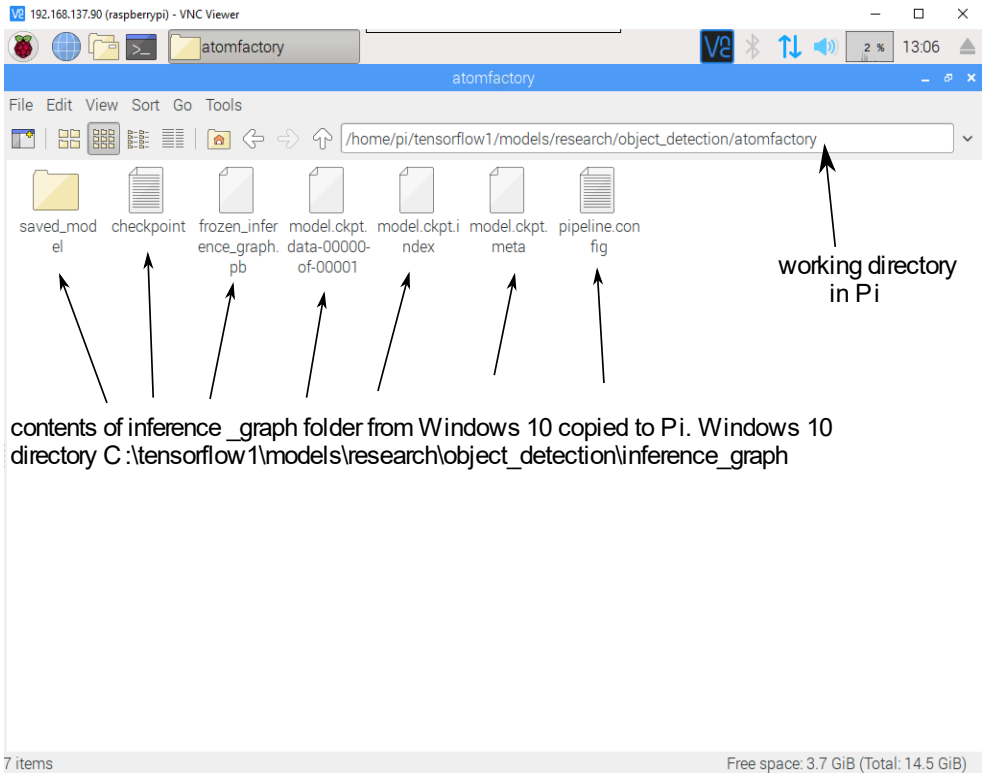



Figure 4-37 Overview of contained files in atom factory folder

Now we are going to create a folder inside Pi's object detection folder. We have named the folder as data. The data folder contains the labelmap.pbtxt file shown in Figure 4-38. It is the same file that we have used to train our object detection classifier. So, we can copy the labelmap.pbtxt from the training folder of Windows 10 to the data folder of Pi.

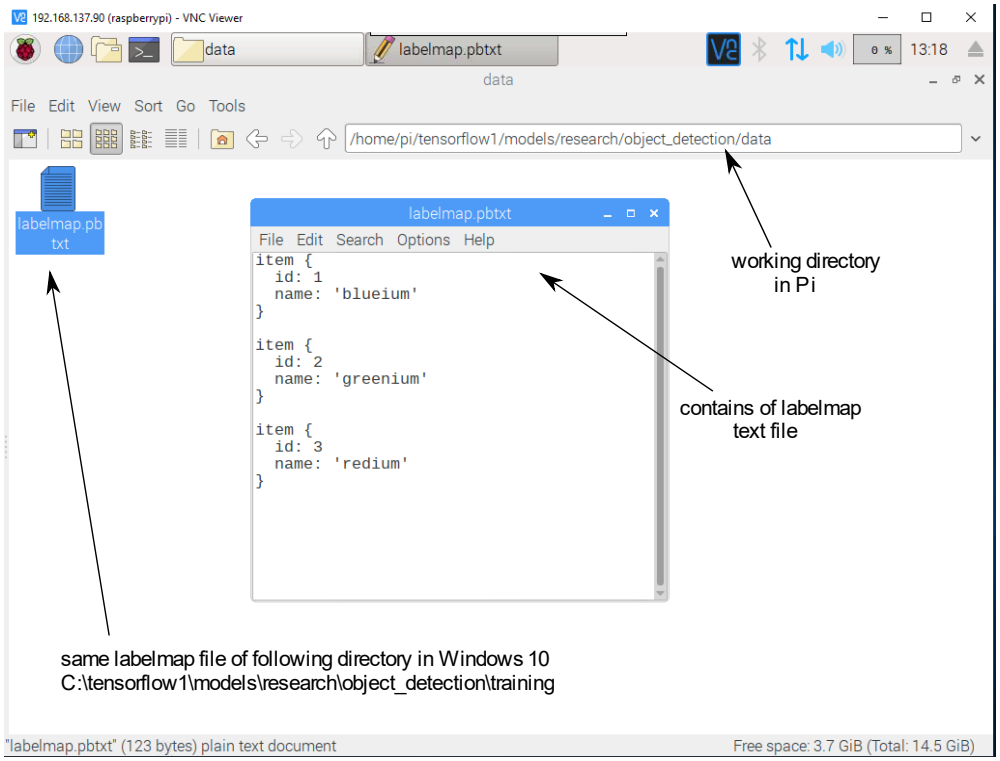


Figure 4-38 Overview of contained files in the data folder

# 4.7. Circuit connection

Now let us discuss the circuit connection we are implementing on the robot. We have already discussed the items and parts we are going to use. We have numbered the connection for better understanding as shown in Figure 4-39. We have used two 6V batteries and connected them in series to get 12V output voltage.

1. Batteries to the LM2956 DC-DC converter

   The batteries get the connection to the LM2956 converter. The power of the battery is connected to the Vin+, and the ground is connected to the Vin- pin of the converter.

2. Batteries to the 5V DC to USB converter

   Here the batteries get the connection to the 5V DC to USB converter. The power of the batteries is connected to the positive terminal, and the ground is connected to the ground terminal of the converter.

3. LM2956 DC-DC converter to L298n Motor drive

   The Vout+ pin of the converter is connected to the +12V input pin of the L298n motor drive and the Vout- point of the converter is connected to the ground pin of L298n.

4. 5V USB converter to Raspberry Pi

   We have used a micro USB cable which connects the 5V USB converter to the Raspberry Pi to power up the Pi.

5. L298n motor drive to Arduino Uno

   The +5V output voltage pin of the L298n motor drive is providing the power to the Arduino Uno. Also, the in1, in2, in3, and in4 pins of the motor drive is connected to the digital pins of the Arduino to command the Motor A and Motor B.

6. L298n Motor drive to Motor A and Motor B

   The plus and minus pin of the Motor A is connected to the out1 and out2 pin of the motor drive. In the case of Motor B, the positive and negative terminal is connected to the out3 and out4 pins of the motor drive.

7. LM2956 DC-DC converter to breadboard

   The output voltage of 7.6V from the L2956 converter has been shorted to the breadboard. The common ground pin on the breadboard is taken from the ground pin of the L298n motor drive. Later we have used the breadboard ground and power connections to power up the servo motors of the mechanical arm.

8. Breadboard to Arduino Uno

   A common ground has been established between the breadboard's ground pin to Arduino Uno's ground pin.

9. Arduino Uno to Raspberry Pi 3 Model B v1.2

   As mentioned earlier, we have implemented I2C communication between Arduino and Raspberry Pi. So, there is a common ground between them and SDA1, SCL1 pins of the Pi is connected to the Analog pins of the Arduino Uno.

10. Breadboard to Raspberry Pi 3 Model B v1.2

    Same as in connection 8, A common ground has been established between the breadboard's ground pin to Pi's ground pin.

11. Servo motors to Breadboard

    Servo motors ground and power pin are connected to the breadboard's common ground pin and +12V power input pin.

12. Arduino Uno to Servo motors

    Control pins of the servo motors of the mechanical arm are connected to the digital pins of the Arduino Uno.

13. Raspberry Pi 3 Model B v1.2 to Pi Camera

    Raspberry Pi is connected to the Pi Camera through flex cable.

.

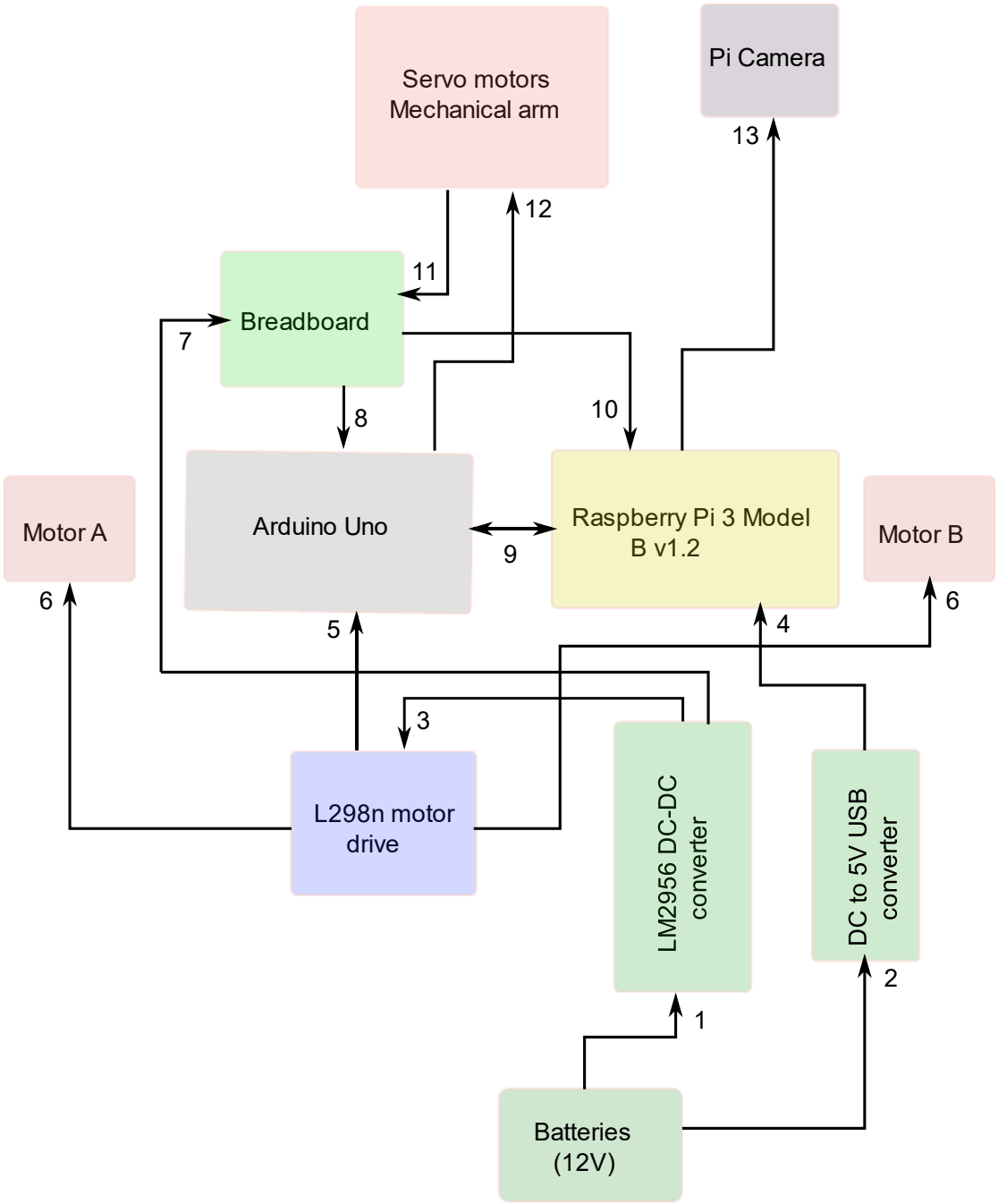



Figure 4-39 Circuit connection diagram

# 4.8. Tuning Mechanical Robot Arm

The opening diameter of the inbuilt jaws of the mechanical arm is not sufficient enough so that the atom can be placed in between them. In other words, the opening of the jaws was very less to put the atom inside as shown in Figue 4-40.

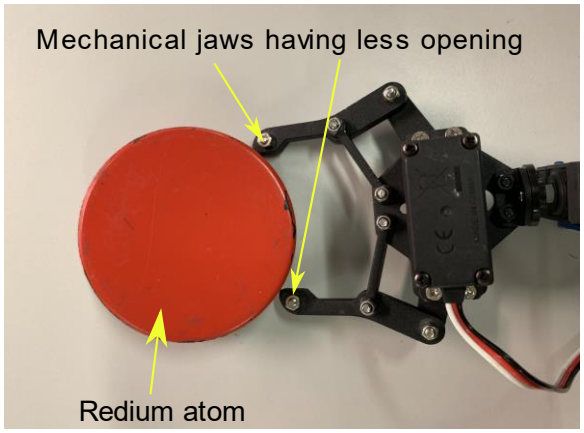
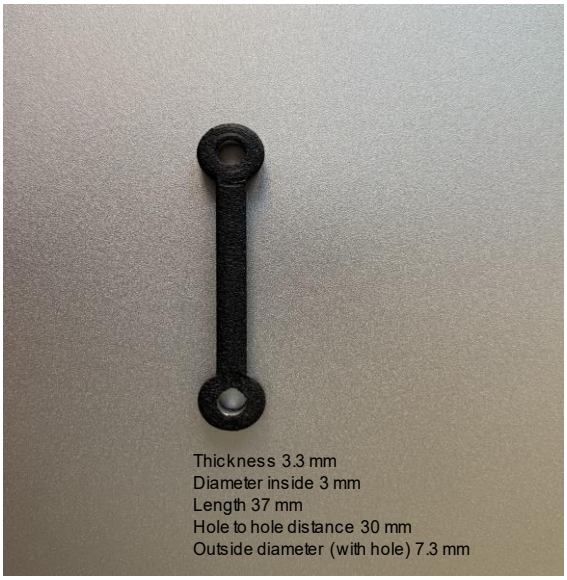



Figure 4-40 Mechanical jaws issue

So first we needed to find a way to increase the size of the opening of the jaws. For that, we have thought about various solutions. The most convenient way was to change the four joints of the arm, which were mainly acting behind the diameter of the jaw opening.

The mechanical part has to be precise in order to integrate with the already built setup. So, we have measured the original part in detail as shown in Figure 4-41.



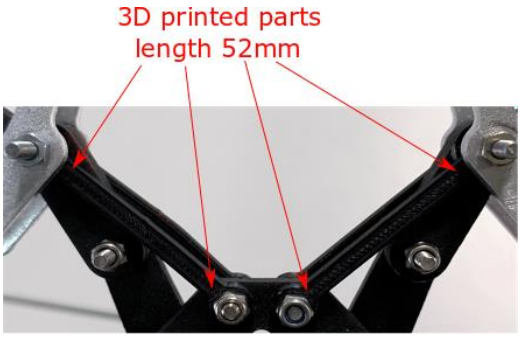




Figure 4-41 Original part on the left, 3D printed four parts on the right

By increasing its length, we can make the jaw to be open more widely. After measuring the joint, we have 3D printed the four joints having a length of 52 mm. This extra length gives us the larger jaw opening as shown in Figure 4-42.

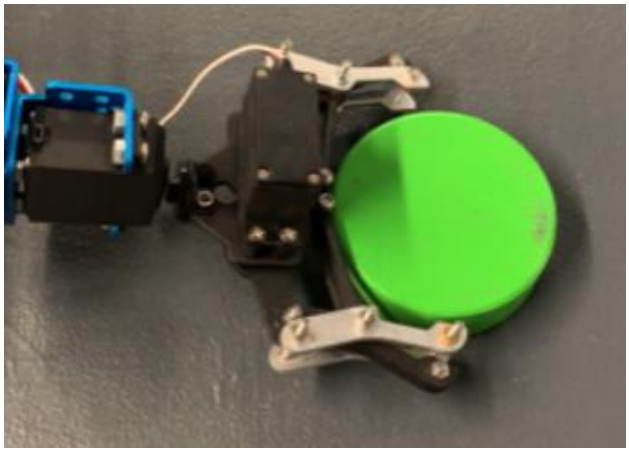


Figure 4-42 Widen the opening of the Jaws

Now we are facing another problem. The in-built mechanical arm structure was able to grab an object. As we have changed its joints, we were unable to make a grabbing movement over the atom. For that, we have developed a plan. As shown in the Figure 4-43 the plan worked.

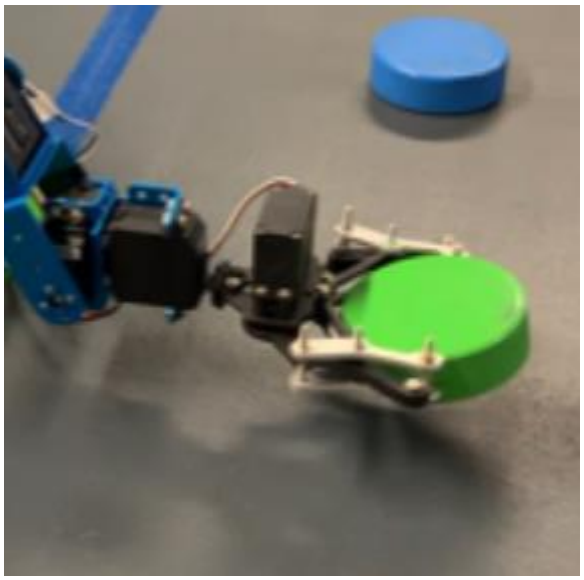


Figure 4-43 Mechanical jaws grab an atom

We have used two other jaw parts and joined them in a way so that our jaws can grab the atom.

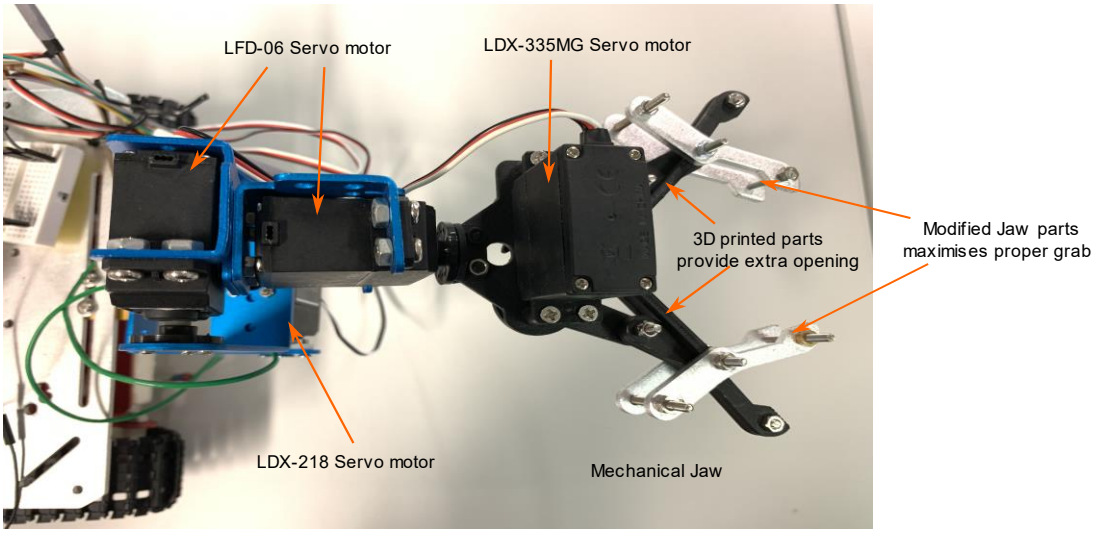



Figure 4-44 Mechanical arm final stage

As a result, now our mechanical robot arm can produce enough opening to let the atom to be inside its jaw as shown in Figure 4-44. Two extra modified jaws are helping to grab and hold the atom properly.

## 4.9. Algorithm for Robot navigation

There are in total of six different scenarios where atoms are placed in front of the periodic tables and the robot has to classify them and put them according to their periodic table. In this thesis, we have developed separate algorithms for each of the scenarios. The starting point of the robot is fixed which is from Greenium periodic table slot for all the scenarios.

Let us consider the two scenarios shown in Figure 4-45.

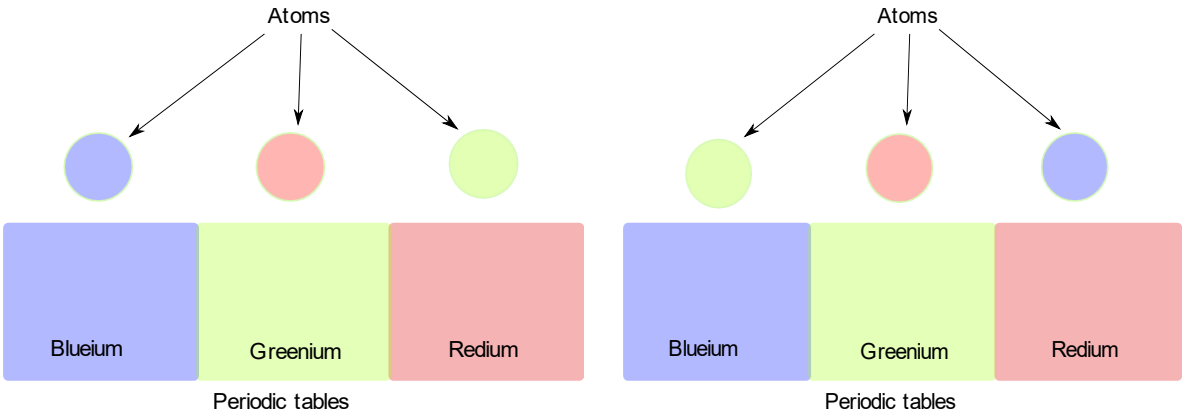



Figure 4-45 Scenario BRG (left figure) and GRB (right figure)

Here, three atoms Blueium, Redium and Greenium are placed in front of Blueium, Greenium, and Redium periodic table. The placement of Redium atom is fixed which is in front of Greenium Periodic table where Greenium and Blueium atoms are in two different positions. The algorithm for robot movement follows.

I. The Robot detect the atom (Send a signal to Arduino from Pi).
II. If the atom is Redium then move forward and grab the atom.
III. Rotate right for 90 degree.
IV. Place Redium atom on the Redium periodic table slot.
V. Rotate left for 40 degree.
VI. The robot starts to detect the atom which is currently in front of the PiCamera.

Now, in case of Greenium atom is in front of Redium periodic table slot as shown in Figure 4-46.

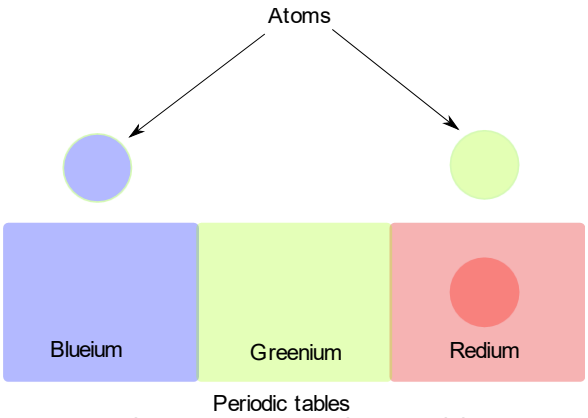


Figure 4-46 Scenario BRG (1)

VII. If the atom is Greenium then grab Greenium atom.
VIII. Move forward and rotate for 180 degree.
IX. Place Greenium on the Greenium periodic table slot (shown in Figure 4-47).

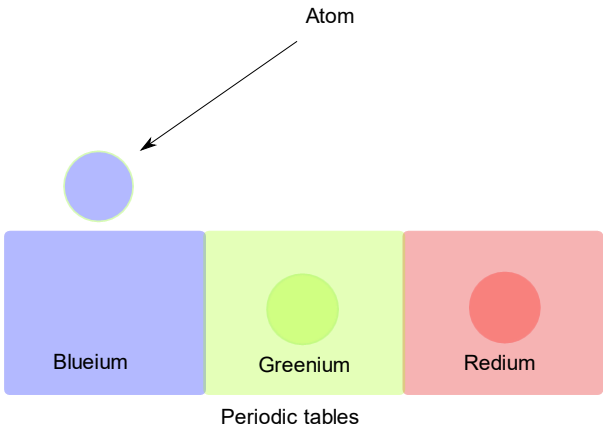


Figure 4-47 Scenario BRG (2)

X. Rotate right for 40 degree.
XI. Grab the Blueium atom.
XII. Rotate left for 40 degree.
XIII. Place the Blueium atom on the Blueium periodic table slot.

In the case of Blueium atom in front of Redium Periodic table slot as shown in Figure 4-48, the algorithm follows.

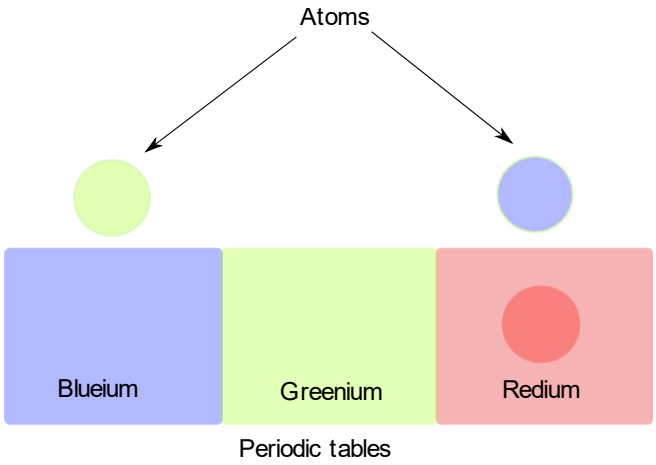



Figure 4-48 Scenario GRB (1)

VII. Grab the Blueium atom.
VIII. Rotate left for 180 degree.
IX. Place the Blueium atom on the Blueium periodic table slot (shown in Figure 4-49).

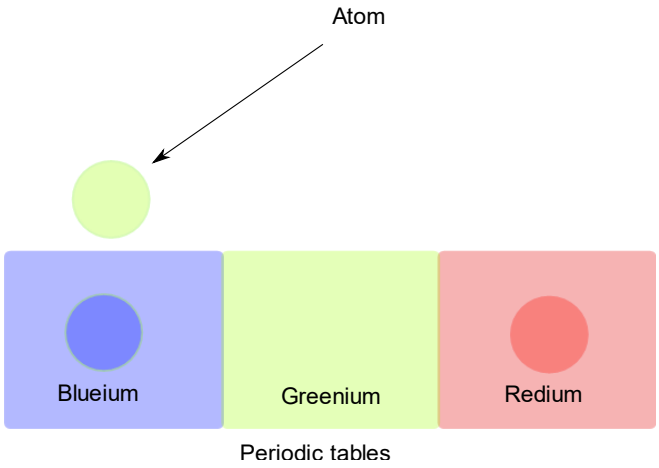



Figure 4-49 Scenario GRB (2)

X. Rotate right for 40 degree.
XI. Grab the Greenium atom.
XII. Move forward and rotate left for 180 degree.
XIII. Place Greenium atom on the Greenium periodic table slot.

Placing the Greenium atom on the Greenium periodic table slot finishes the goal for that particular scenario. Now, let us consider another two scenarios shown in Figure 4-50 below

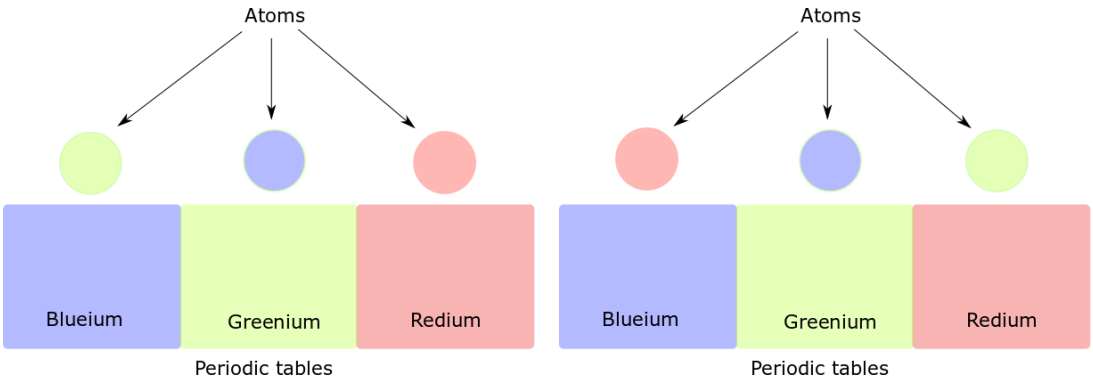



Figure 4-50 Scenario GBR (left figure) and RBG (right figure)

The placement of Blueium atom here is fixed which is in front of Greenium Periodic table where Greenium and Redium atoms are in two different positions. The algorithm follows

I. The robot detects the atom (Send a signal to Arduino from Pi).
II. If the atom is Blueium then move forward and grab the atom.
III. Rotate left for 90 degree.
IV. Place the Blueium atom on the Blueium periodic table slot.
V. Rotate right for 40 degree and robot starts to detect the atom which is currently in front of the PiCamera.

Now, in case of Greenium atom is in front of Blueium periodic table slot as shown in Figure 4-51.

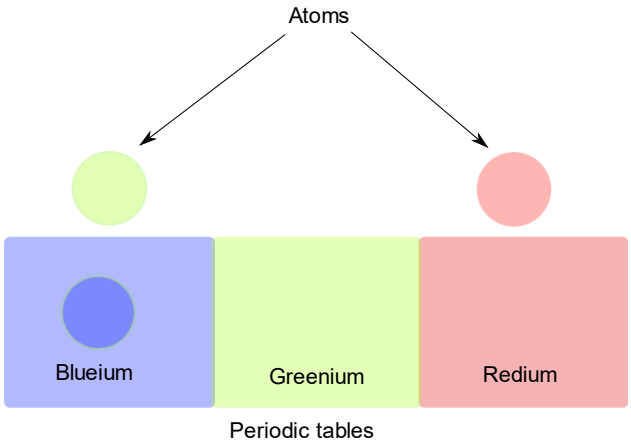



Figure 4-51 Scenario GBR (1)

VI. If the atom is Greenium then grab Greenium atom.
VII. Rotate 180 for degree holding the Greenium atom.
VIII. Move backwards and place the Greenium atom on the Greenium periodic table slot (shown in Figure 4-52).

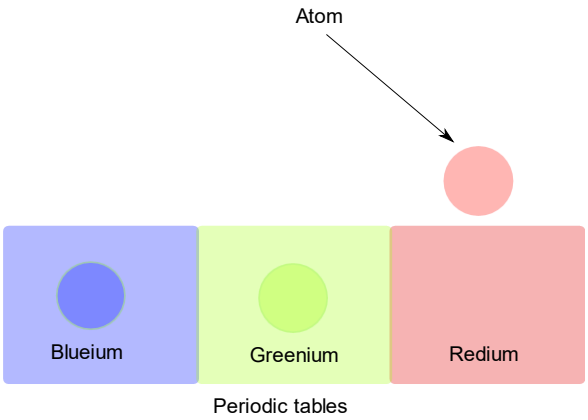



Figure 4-52 Scenario GBR (2)

IX. Rotate left for 40 degree.
X. Grab the Redium atom.
XI. Rotate right for 40 degree.
XII. Place the Redium atom on the Redium periodic table slot.

In the case of Redium atom is in front of Blueium periodic table slot (shown in Figure 4-53), the algorithm follows

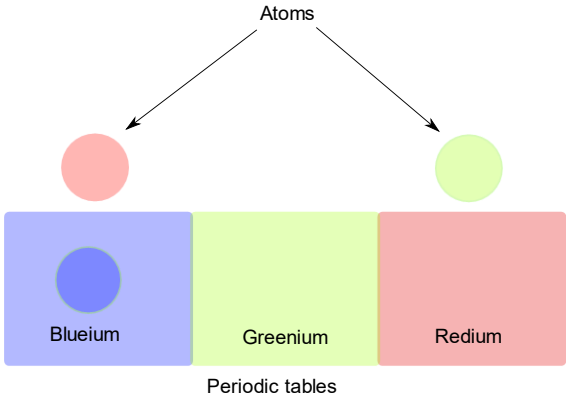



Figure 4-53 Scenario RBG (1)

VI. If the atom is Redium then grab the Redium atom.
VII. Rotate left for 180 degree and move forward.
VIII. Place Redium atom on the Redium periodic table slot (shown in Figure 4-54).
IX. Move backwards left for 40 degree.

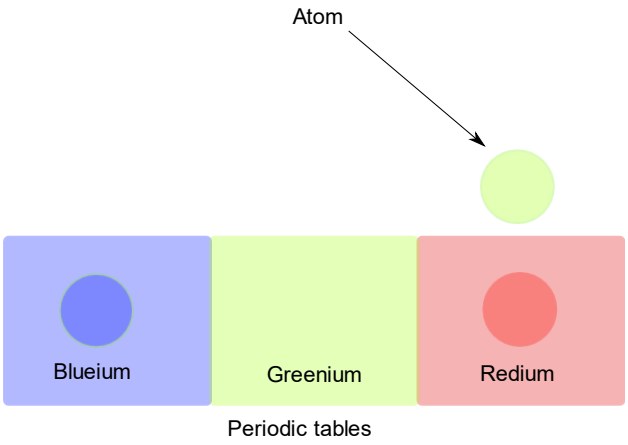



Figure 4-54 Scenario RBG (2)

X. Grab Greenium atom from the front of Redium periodic table slot.
XI. Move backwards right for 90 degree.
XII. Place Greenium atom on the Greenium periodic table slot.

Now let us consider the last two scenarios shown in Figure 4-55 below.

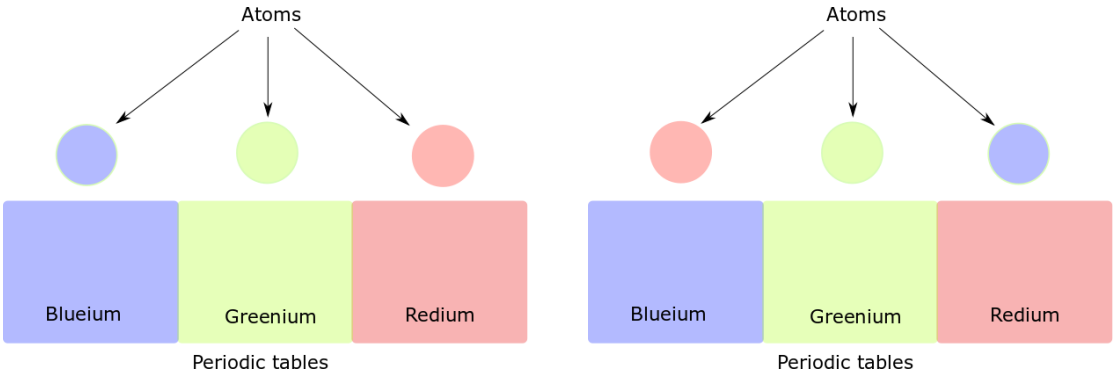



Figure 4-55 Scenario BGR (left figure) and RGB (right figure)

Here, the placement of the Greenium atom is fixed which is in front of the Greenium Periodic table while the Blueium and Redium atoms are in two different positions. The algorithm follows

I. The robot detects the atom (Send a signal to Arduino from Pi).
II. If the atom is Greenium then move forward and grab the atom.
III. Move forward and rotate left for 180 degree.
IV. Move forward and place Greenium atom on the Greenium periodic table slot.
V. Move backwards right for 40 degree.
VI. The robot starts to detect the atom which is currently in front of the PiCamera.

Considering the Blueium atom is in front of the Blueium periodic table slot (shown in Figure 4-56), the algorithm follows as

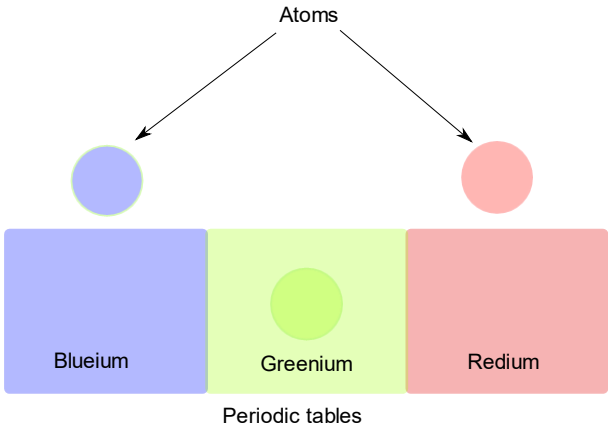



Figure 4-56 Scenario BGR (1)

VII. If the atom is Blueium then grab the atom.
VIII. Rotate left for 60 degree.
IX. Put the Blueium atom on the Blueium periodic table slot as shown in Figure 4-57.

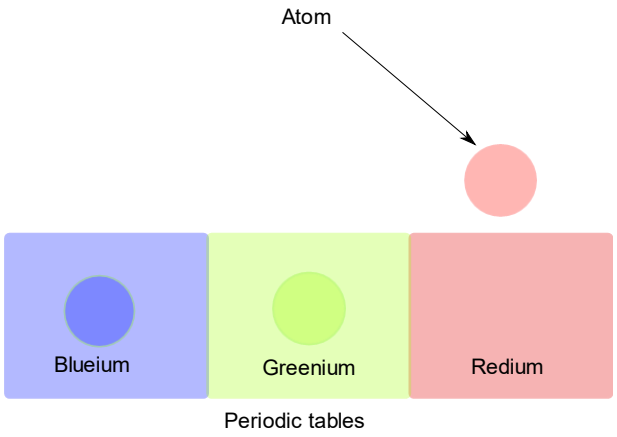



Figure 4-57 Scenario BGR (2)

X. Rotate left for 90 degree.
XI. Move forward and grab the Redium atom from in front of the Redium periodic table slot.
XII. Rotate right for 90 degree.
XIII. Place the Redium atom on the Redium periodic table slot.

Now, considering the Redium atom is in front of the Blueium periodic table slot (shown in Figure 4-58), the algorithm step changes as

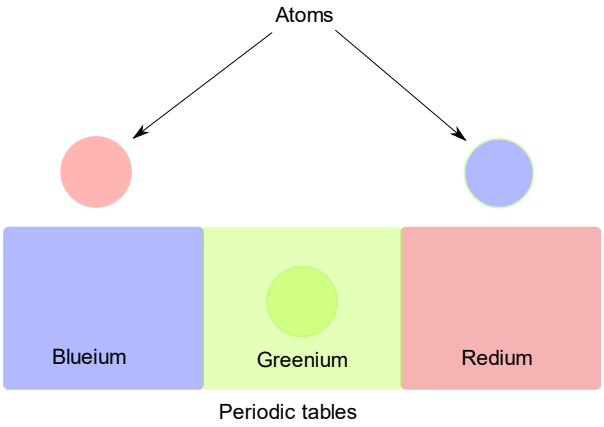



Figure 4-58 Scenario RGB (1)

VII. If the atom is Redium then grab the atom.
VIII. Rotate left for 140 degree and move forward.
IX. Place the Redium atom on the Redium periodic table slot (shown in Figure 4-59).

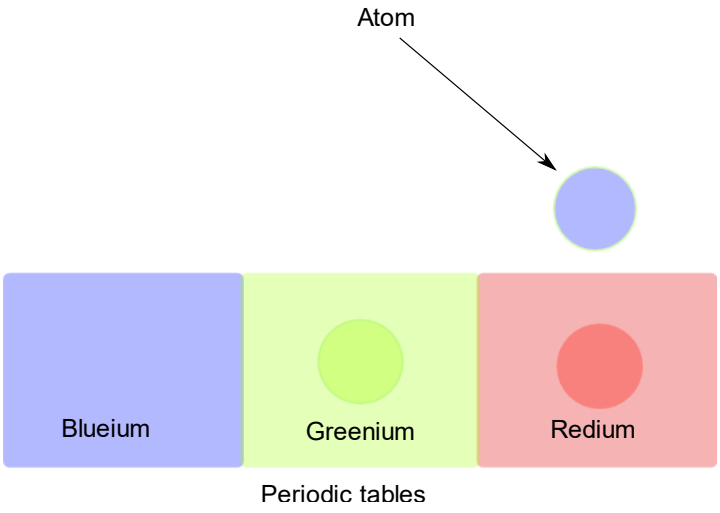



Figure 4-59 Scenario RGB (2)

X. Backwards rotate left for 40 degree
XI. Grab the Blueium atom from in front of the Redium periodic table slot
XII. Rotate left for 180 degree
XIII. Place the Blueium atom on the Blueium periodic table slot

## 4.10. Algorithm for PiCamera object detection using TensorFlow

The algorithm follows the following points.

- Load the frozen inference graph shown in Figure 4-60.

```
# Name of the directory containing the object detection module we're using
MODEL_NAME = 'atomfactory'

# Grab path to current working directory
CWD_PATH = os.getcwd()

# Path to frozen detection graph .pb file, which contains the model that is used
# for object detection.
PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')
```

Figure 4-60 Loading the frozen graph

- Load the TensorFlow model in the memory shown in Figure 4-61.

```
# Load the Tensorflow model into memory.
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

    sess = tf.Session(graph=detection_graph)
```

Figure 4-61 Loading the model in the system memory

- Start streaming video and run the object detection classifier on the video to detect the trained objects from the video shown in Figure 4-62 and Figure 4-63.

```
if camera_type == 'picamera':
    # Initialize Picamera and grab reference to the raw capture
    camera = PiCamera()
    camera.resolution = (IM_WIDTH,IM_HEIGHT)
    camera.framerate = 10
    rawCapture = PiRGBArray(camera, size=(IM_WIDTH,IM_HEIGHT))
    rawCapture.truncate(0)
    time.sleep(2) #test
    for frame1 in camera.capture_continuous(rawCapture, format="bgr",use_video_port=True):

        t1 = cv2.getTickCount()

        # Acquire frame and expand frame dimensions to have shape: [1, None, None, 3]
        # i.e. a single-column array,. where each item in the column has the pixel RGB value
        frame = np.copy(frame1.array)
        frame.setflags(write=1)
        frame_expanded = np.expand_dims(frame, axis=0)

        # Perform the actual detection by running the model with the image as input
        (boxes, scores, classes, num) = sess.run(
            [detection_boxes, detection_scores, detection_classes, num_detections],
            feed_dict={image_tensor: frame_expanded})

        # Draw the results of the detection (aka 'visulaize the results')
        vis_util.visualize_boxes_and_labels_on_image_array(
            frame,
            np.squeeze(boxes),
            np.squeeze(classes).astype(np.int32),
            np.squeeze(scores),
            category_index,
            use_normalized_coordinates=True,
            line_thickness=8,
            min_score_thresh=0.40)
```

Figure 4-62 Drawing the boundary box (1)

```
cv2.putText(frame,"FPS: {0:.2f}".format(frame_rate_calc),(30,50),font,1,(255,255,0),2,cv2.LINE_AA)
```

Figure 4-63 Drawing the boundary box (2)

- If the object is detected for 5 frames, send a signal from Pi to the Arduino and go to sleep for 45 seconds shown in Figure 4-64.

```
if (int(classes[0][0]) == 1):
    counter_b = counter_b + 1
    #start = time.time()
    if counter_b > 5:
        value = int(1)
        writeNumber(value)
        #if time.time() - start > 20:
        print ('Blueium is detected in picam')

        # sleep
        time.sleep(45)
```

Figure 4-64 Sending signal to Arduino and sleep

- After 45 seconds the Pi Camera starts to do the object detection on the live stream video again.

# 4.11. Developed robot

The Figure 4-65, 4-66 and 4-67 shows the robot that we have developed in this master thesis from three different viewpoints.

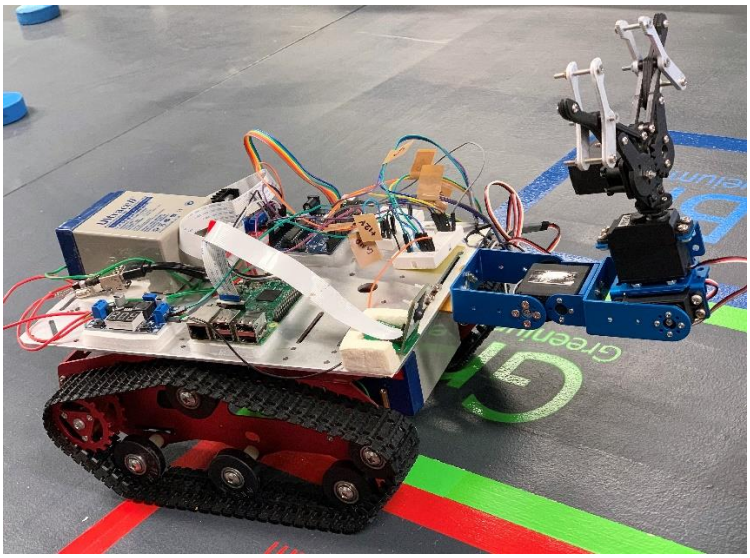


Figure 4-65 Developed robot sideview

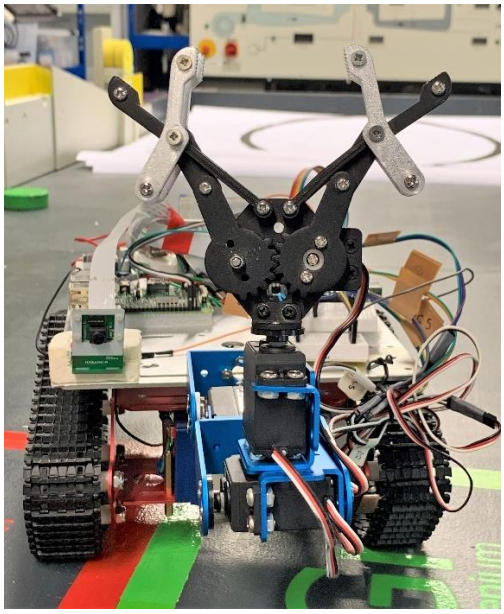


Figure 4-66 Developed robot front view

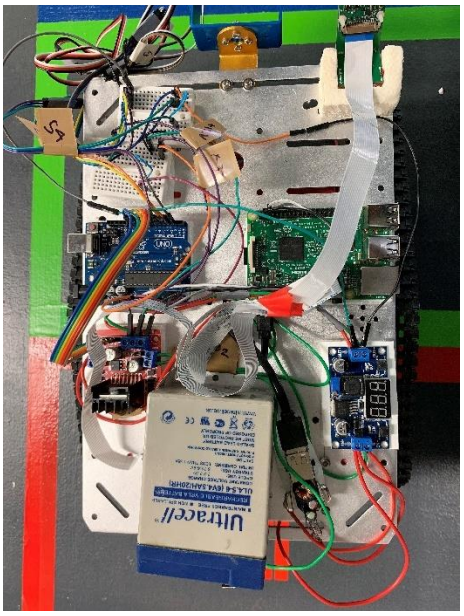



Figure 4-67 Developed robot top view

# Chapter 5 Results

We are following the algorithm mentioned earlier in the implementation chapter. Let us consider the scenario RBG first.

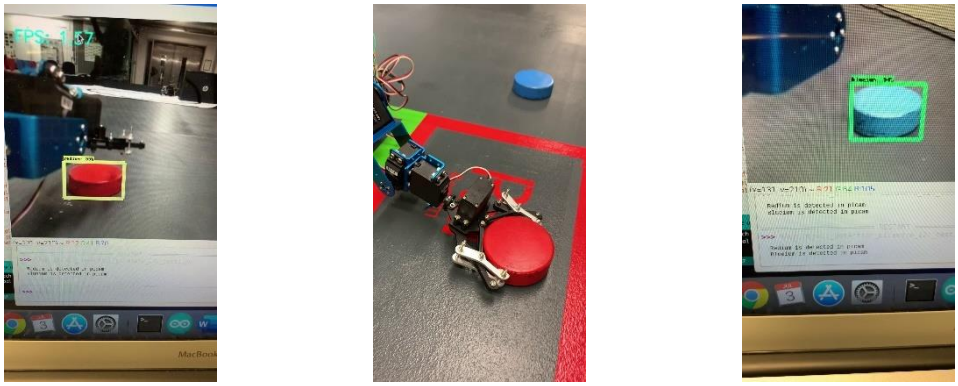


Figure 5-1 RBG scenario (1)

In the Figure 5-1, from the left, the Pi camera is detecting the atom and sending the signal to the Arduino for the navigation. Then the mechanical arm is placing the Redium atom on the Redium periodic table slot. After that, Pi camera is starting to detect the atom which is currently in front of the camera.

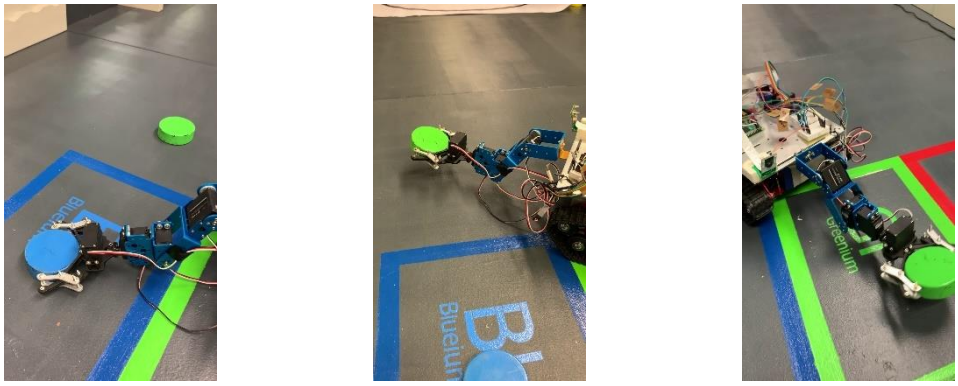


Figure 5-2 RBG scenario (2)

In the Figure 5-2, from the left, the robot is placing the Blueium atom on the Blueium periodic table slot and then rotate 40 degree right and grab the Greenium atom and finally placing the Greenium atom on the Greenium periodic table slot.

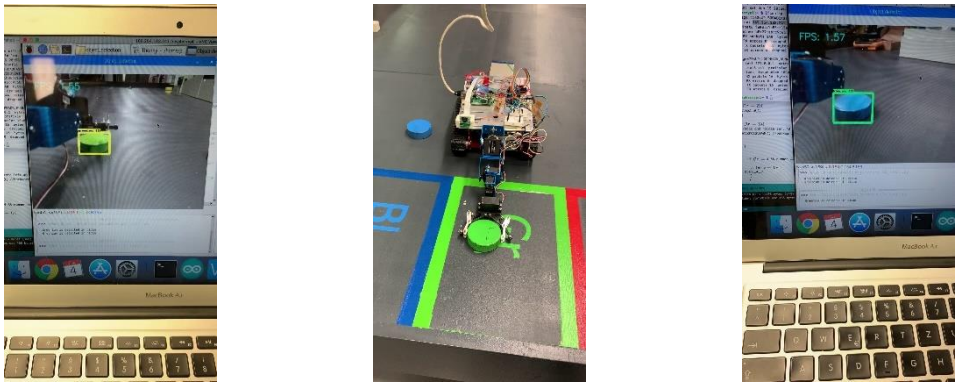


Figure 5-3 GBR scenario (1)

Now let us consider the scenario GBR. In the Figure 5-3, from the left, the Pi camera is detecting the atom and sending the signal to the Arduino for the navigation. Then the mechanical arm is placing the Greenium atom on the Greenium periodic table slot. After that, Pi camera is starting to detect the atom which is currently in front of the camera.

Figure 5-4 GBR scenario (2)

In the Figure 5-4, from the left, the robot is placing the Blueium atom on the Blueium periodic table slot and then rotate left 90 degree and grab the Redium atom and finally placing the Redium atom on the Redium periodic table slot.

Figure 5-5 RGB scenario (1)

In the case of RGB (shown in Figure 5-5) , from the left, the Pi camera is detecting the atom and sending the signal to the Arduino for the navigation. Then the mechanical arm is placing

the Redium atom on the Redium periodic table slot. After that, Pi camera is starting to detect the atom which is currently in front of the camera.

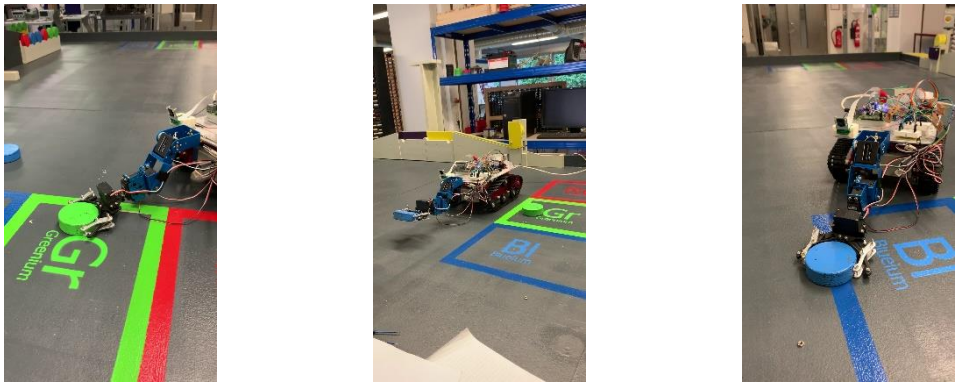


Figure 5-6 RGB scenario (2)

In the Figure 5-6, from the left, the robot is placing the Greenium atom on the Greenium periodic table slot and then rotate right 40 degree and grab the Blueium atom and finally placing the Blueium atom on the Blueium periodic table slot.

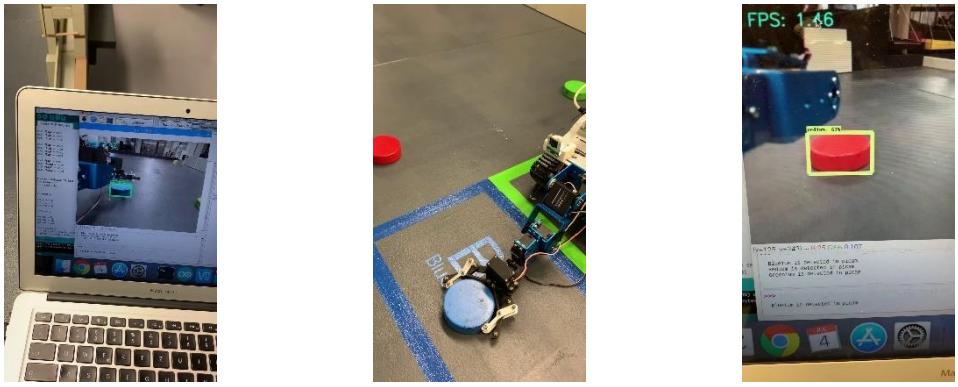



Figure 5-7 BRG scenario (1)

In the case of BRG shown in Figure 5-7, from the left, the Pi camera is detecting the atom and sending the signal to the Arduino for the navigation. Then the mechanical arm is placing the Blueium atom on the Blueium periodic table slot. After that, Pi camera is starting to detect the atom which is currently in front of the camera.

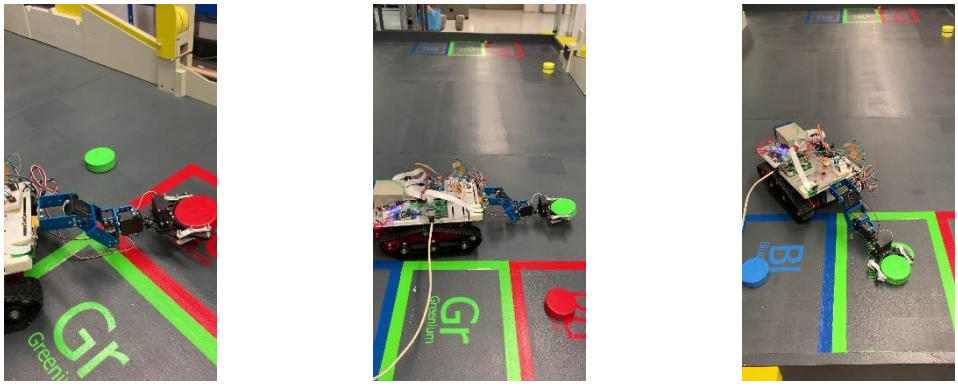


Figure 5-8 RGB scenario (2)

In the Figure 5-8, from the left, the robot is placing the Redium atom on the Greenium periodic table slot and then rotate left 40 degree and grab the Greenium atom and finally placing the Greenium atom on the Greenium periodic table slot.

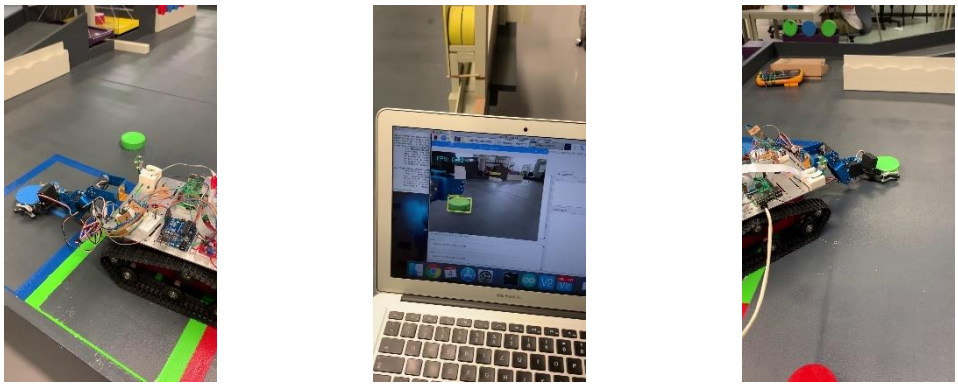


Figure 5-9 BRG scenario (1)

In the case of BGR scenario shown in Figure 5-9, from the left, the Pi camera is detecting the atom and sending the signal to the Arduino for the navigation. Then the mechanical arm is placing the Blueium atom on the Blueium periodic table slot. After that, Pi camera is starting to detect the atom which is currently in front of the camera.

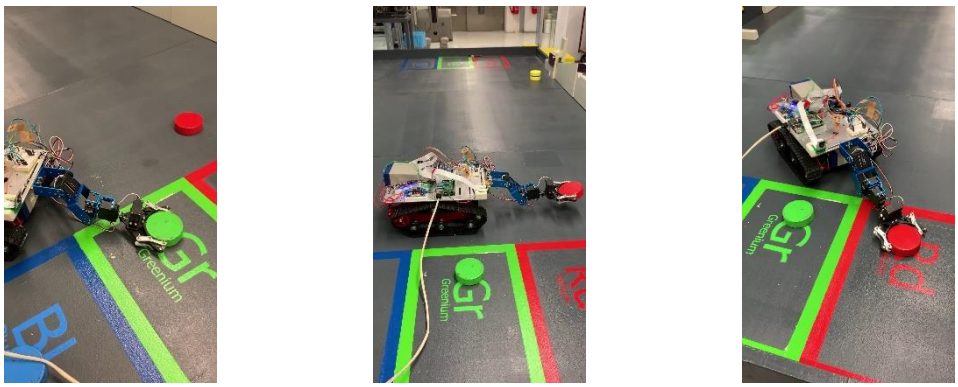


Figure 5-10 BRG scenario (2)

In the Figure 5-10, from the left, the robot is placing the Greenium atom on the Greenium periodic table slot and then rotate left 40 degree and grab the Redium atom and finally placing the Redium atom on the Redium periodic table slot.

Same as the scenarios shown above, the robot can do its corresponding steps in case of GRB. Unfortunately, we were unable to take proper video documentation regarding this scenario. But the algorithm is working and the robot is performing just like other scenarios.

# Chapter 6 Discussion

## 6.1. General Discussion

### 6.1.1 Robot Chassis and Mechanical Robot Arm

The robot chassis and mechanical robot arm that we have used in the thesis was unassembled and were available in our mechanical lab of the SRH Hochschule Heidelberg. Those parts were already have been used in other projects so we had to make a lot of adjustments. As many of the unassembled parts were available, it took us not much longer time to come up with a robot chassis and mechanical robot arm. After that, we have started adjusting the chassis and the arm as our needs for executing the mission.

### 6.1.2 Spending

For this thesis, all the materials and types of equipment were provided by the mechanical lab of the Architecture and Engineering Department of SRH Hochschule Heidelberg. The mechanical lab is enriched with all levels of hardware materials. For the thesis, we have 3D printed four joints of the mechanical jaws from the 3D printer available at the mechanical lab.

From my own expense, I have brought a DC-DC L2956 converter after blowing up one as there was not another workable converter available at that time. Another electronic part I had to purchase which is the DC to 5V USB converter to power up the Raspberry Pi using the DC batteries as there was no power converter available in the lab for the Pi on that time being. In total, it cost me around 13 euro for both of the electric parts.

### 6.1.3 Notable problems

The first notable problem we have faced during the thesis is to establish an ssh connection to the Pi from Windows10 using the ethernet cable. We have already discussed this problem in the implementation chapter. The second notable problem was with the mechanical arm. As the mechanical arm was unable to widen up the jaws different approach had to be taken. After widen up the mechanical jaws opening, we have faced the problem to hold the atom inside the jaws. For that, we have added two mechanical jaws in a modified way.

After the mechanical arm part is done, we were facing problem regarding the power issue of the servo motors. As in the original game elements in the Eurobot2019, game elements have specific weight. In our case, the atoms we are dealing with are much heavier. Because of that, one of the servo motors (LDX-218) has to overperform in every running time of the robot. It is advised not to run the servo motors continuously for better service life. So, we needed to keep an eye on the temperature of the LDX-218 servo motor.

Due to applying the power (12V) much more than the recommended power (7.5V) on the servo motors, we had to change the servo motors we have been using from the beginning of the thesis. Most recently, due to miswiring the positive and negative terminal, the DC-DC L2596 converter got burned then we had to get a new converter.

Managing the power is another major problem we are facing regarding executing the steps. As the batteries we are using to run the robot gets discharged, the functionalities that we have placed are not executing as our plan all the time. The delays we have given to the robot's motors get change according to the power. As the robot is running on the battery, the power is deteriorating which ultimately changes the robot's movement speed and precision. For that reason, we need to monitor the batteries charge level and, in some cases, adjust the delays for successfully performing the task

One of the most challenging problems was to power up the Raspberry Pi without using a power bank or extra battery. As mentioned in the earlier chapter, Raspberry Pi needs specific voltage and current to run properly which is around 5.25V and 2.5 Amps. As we are using device resource demanding TensorFlow in Pi, we must need to apply the steady current of 2.5 Amps.

Currently, the problem we are facing is, we have developed all the algorithms in one program file. But they are not running concurrently which means we have to upload the code to the Arduino according to the scenario rather than one program which automatically does the steps according to the scenario.

## 6.2. Conclusion

The main goal of the thesis was to develop a budget efficient robot that performs the first mission of Eurobot2019. Instead of going for advanced equipment like laser sensors, LiDAR or beacon beams, here we have tried to implement the mission with the hardware available in the mechanical lab of our university. Instead of traditional image processing algorithm, which takes a vast programming knowledge to implement, we have used Google's TensorFlow object detection API's which uses deep learning algorithm. As for the conclusion, we have been successfully developed an autonomous robot which can perform the first task of Eurobot2019. Still, we have to a long way to go to execute the mission in a more efficient manner.

## 6.3. Future work

There are many open ends regarding the navigation system that we have implemented in this thesis. As we have hardcoded the movements of the robots, there are more efficient ways and improvements to do for the navigation purpose and performance enhancement. They are mentioned below

- Real-time robot navigation using the Pi Camera and machine learning.
- Using LiDAR for navigation.
- Using a laser sensor to measure the distance between the object and the robot.
- Use more suitable servo motors for handling the weight load with ease.
- Use high precision DC motors with advance motor encoder circuitry.
- Use a beacon support system for robot navigation.
- Use of more suitable battery source like lithium-ion polymer battery for more steady power discharge.
- Instead of using factory made power supply converter for Pi, develop own power supply converter.

## 6.4. Listing

All the codes that are used in this thesis paper are uploaded in the following github repository.

```
https://github.com/ayonfahad/Master_thesis_robot_development
```

# Notation and Abbreviations

| | |
|---|---|
| DC | Direct Current |
| PWM | Pulse Width Modulation |
| SCL | Serial Clock Line |
| SDA | Serial Data Line |
| V | Voltage |
| VNC | Virtual Network Console |
| USB | Universal Serial Bus |
| GND | Ground |
| GUI | Graphical User Interface |
| $I^2C$ | I-squared-C |
| VDC | volts DC |
| enA | Enable Motor A |
| enB | Enable Motor B |
| SoC | System on a Chip |
| RISC | Reduced Instruction Set Computing |
| CISC | Complex Instruction Set Computing |
| CPI | Cycle Per Instruction |
| ARM | Advanced RISC Machine |
| IoT | Internet of Things |
| OS | Operating System |
| CPU | Central Processing Unit |
| GPU | Graphics Processing Unit |
| HDMI | High Definition Multimedia Interface |
| MB | Mega Byte |
| GB | Giga Byte |
| SD | Secure Digital |
| IO | Input Output |
| GPIO | General Purpose Input Output |
| WLAN | Wireless Local Area Network |
| GB | Giga Byte |
| LPDDR | Low-Power Double Data Rate Synchronous Dynamic Random Access Memory |
| RAM | Random Access Memory |
| PCI-E | Peripheral Component Interconnect Express |
| 802.11bgn | Frequency band of b/g/n (2.5/5 GHz) |
| MP4 | MPEG-4 Part 14 |
| MPEG | Motion Picture Expert Group |
| BLE | Bluetooth Low Energy |
| Amps | Amperes |
| SPI | Serial Peripheral Interface |
| CD | Compact Disk |
| GPU | Graphics Processing Unit |
| SRAM | Static Random Access Memory |
| g | Gram |
| mm | Millimetre |

| | |
|---|---|
| EEPROM | Electrically Erasable Programmable Read-Only Memory |
| IP | Internet Protocol |
| VNC | Virtual Network Computing |
| RFB | Remote Frame Buffer |
| VOC | Visual Object Classes |
| BGR | Blueium Greenium Redium |
| RGB | Redium Greenium Blueium |
| BRG | Blueium Redium Greenium |
| GRB | Greenium Redium Blueium |
| RBG | Redium Blueium Greenium |
| GBR | Greenium Blueium Redium |
| LiDAR | Light Detection and Ranging |
| XML | Extensible Markup Language |
| CSV | Comma Separated Value |

# List of Figures

# List of Tables

# Bibliography


[1] S. Yegulalp, „What is TensorFlow? The machine learning library explained," 2019. [Online]. Available: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html, last accessed on 2019-07-05.

[2] Wikipedia, „TensorFlow," [Online]. Available: https://en.wikipedia.org/wiki/TensorFlow, last accessed on 2019-07-05.

[3] SFUPTOWNMAKER, „I2C," [Online]. Available: https://learn.sparkfun.com/tutorials/i2c last accessed on 2019-07-05.

[4] pkulzc, „Tensorflow detection model zoo," [Online]. Available: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md, last accessed on 2019-07-05.

[5] eResearch, „Create virtual environments for python with conda," 2014. [Online]. Available: https://uoa-eresearch.github.io/eresearch-cookbook/recipe/2014/11/20/conda/, last accessed on 2019-07-05.

[6] E. Electronics, „Tutorial to set up TensorFlow Object Detection API on the Raspberry Pi," [Online]. Available: https://github.com/EdjeElectronics/TensorFlow-Object-Detection-on-the-Raspberry-Pi, last accessed on 2019-07-05.

[7] E. Electronics, „TensorFlow-Object-Detection-on-the-Raspberry-Pi/Pet_detector.py at master · EdjeElectronics/TensorFlow-Object-Detection-on-the-Raspberry-Pi - Pet_detector," [Online]. Available: https://github.com/EdjeElectronics/TensorFlow-Object-Detection-on-the-Raspberry-Pi/blob/master/Pet_detector.py, last accessed on 2019-07-05.

[8] E. Electronics, „TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10: How to train a TensorFlow Object Detection Classifier for multiple object detection on Windows," [Online]. Available: https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10, last accessed on 2019-07-05..

[9] E. Electronics, „How To Train an Object Detection Classifier Using TensorFlow 1.5 (GPU) on Windows 10 - YouTube," [Online]. Available: https://www.youtube.com/watch?v=Rgpfk6eYxJA, last accessed on 2019-07-05.

[10] E. Electronics, „How to Set Up TensorFlow Object Detection on the Raspberry Pi - YouTube," [Online]. Available: https://www.youtube.com/watch?v=npZ-8Nj1YwY, last accessed on 2019-07-05.

[11] B_E_N, „What is an Arduino?," [Online]. Available: https://learn.sparkfun.com/tutorials/what-is-an-arduino/whats-on-the-board, last accessed on 2019-07-05.

[12] J.-L. AUFRANC, „Raspberry Pi 3 Model B Board Features a 64-Bit ARM Processor, Adds WiFi and Bluetooth Connectivity," 2016. [Online]. Available: https://www.cnx-software.com/2016/02/27/raspberry-pi-3-model-b-board-adds-wifi-and-bluetooth-connectivity/, last accessed on 2019-07-05.

[13] „What is Protocol Buffers used for?," [Online]. Available: https://www.quora.com/What-is-Protocol-Buffers-used-for, last accessed on 2019-07-05.

[14] „TensorFlow," [Online]. Available: https://en.wikipedia.org/wiki/TensorFlow, last accessed on 2019-07-05.

[15] „Servomotor,“ [Online]. Available: https://en.wikipedia.org/wiki/Servomotor last accessed on 2019-07-05.

[16] „Servo Motors| Types, Properties, Control,“ [Online]. Available: http://www.robotiksistem.com/servo_motor_types_properties.html, last accessed on 2019-07-05.

[17] „Reduced instruction set computer,“ [Online]. Available: https://en.wikipedia.org/wiki/Reduced_instruction_set_computer, last accessed on 2019-07-05.

[18] „Raspberry Pi Camera Guide,“ [Online]. Available: http://raspberrypiguide.de/howtos/raspberry-pi-camera-how-to, last accessed on 2019-07-05..

[19] „Raspberry Pi,“ [Online]. Available: https://en.wikipedia.org/wiki/Raspberry_Pi, last accessed on 2019-07-05.

[20] „Python - difference between command prompt and anaconda prompt,“ [Online]. Available: https://stackoverflow.com/questions/37993175/difference-between-command-prompt-and-anaconda-prompt, last accessed on 2019-07-05.

[21] „Protocol Buffers,“ [Online]. Available: https://en.wikipedia.org/wiki/Protocol_Buffers.

[22] „Programming language,“ [Online]. Available: https://en.wikipedia.org/wiki/Programming_language, last accessed on 2019-07-05.

[23] „Periodic table,“ [Online]. Available: https://en.wikipedia.org/wiki/Periodic_table, last accessed on 2019-07-05.

[24] „Operating system,“ [Online]. Available: https://en.wikipedia.org/wiki/Operating_system.

[25] „Mendeleev's Periodic Table,“ [Online]. Available: https://corrosion-doctors.org/Periodic/Periodic-Mendeleev.html, last accessed on 2019-07-05.

[26] „lobot ldx-218 17kg large torque metal gear dual shaft digital servo,“ [Online]. Available: https://www.banggood.com/LOBOT-LDX-218-17KG-Large-Torque-Metal-Gear-Dual-Shaft-Digital-Servo-p-1152553.html?cur_warehouse=CN, last accessed on 2019-07-05.

[27] „LewanSoul Mechanical claw,“ [Online]. Available: http://www.lewansoul.com/product/detail-6.html, last accessed on 2019-07-05.

[28] „LDX 335MG Digital Metal Servo 180 Degrees/ 19kg Torque,“ [Online]. Available: https://www.aliexpress.com/item/32753124415.html.

[29] „Introduction to Tensor with Tensorflow,“ [Online]. Available: https://www.geeksforgeeks.org/introduction-tensor-tensorflow/, last accessed on 2019-07-05.

[30] „Intel® Optimization for TensorFlow* Installation Guide,“ [Online]. Available: https://software.intel.com/en-us/articles/intel-optimization-for-tensorflow-installation-guide#Anaconda_main_win, last accessed on 2019-07-05.

[31] „Instruction set architecture,“ [Online]. Available: https://en.wikipedia.org/wiki/Instruction_set_architecture, last accessed on 2019-07-05.

[32] „GitHub - EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10: How to train a TensorFlow Object Detection Classifier for multiple object detection on Windows,“ [Online]. Available: https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10, last accessed on 2019-07-05.

[33] „Full Metal Robotic Arm,“ [Online]. Available: https://littlerobotshop.com/product/full-metal-robotic-arm/, last accessed on 2019-07-05.

[34] „FAQs - Raspberry Pi Documentation,“ [Online]. Available: https://www.raspberrypi.org/documentation/faqs/, last accessed on 2019-07-05.

[35] „Eurobot2019_Rules_Cup_OFFICIAL_EN“, last accessed on 2019-07-05.

[36] „Dmitri Mendeleev,“ [Online]. Available: https://en.wikipedia.org/wiki/Dmitri_Mendeleev, last accessed on 2019-07-05.

[37] „Debian,“ [Online]. Available: https://en.wikipedia.org/wiki/Debian, last accessed on 2019-07-05.

[38] „COCO - Common Objects in Context,“ [Online]. Available: http://cocodataset.org/#home, last accessed on 2019-07-05.

[39] „Camera Hardware — Picamera 1.12 documentation,“ [Online]. Available: https://picamera.readthedocs.io/en/release-1.12/fov.html, last accessed on 2019-07-05.

[40] „Basic Recipes — Picamera 1.10 documentation,“ [Online]. Available: https://picamera.readthedocs.io/en/release-1.10/recipes1.html, last accessed on 2019-07-05.

[41] „ARM architecture,“ [Online]. Available: https://en.wikipedia.org/wiki/ARM_architecture, last accessed on 2019-07-05.

[42] „Arduino Uno Rev3,“ [Online]. Available: https://store.arduino.cc/arduino-uno-rev3, last accessed on 2019-07-05.

[43] „Arduino - Software,“ [Online]. Available: https://www.arduino.cc/en/Main/Software, last accessed on 2019-07-05.

[44] „Arduino - Introduction,“ [Online]. Available: https://www.arduino.cc/en/Guide/Introduction last accessed on 2019-07-05.

[45] „Anaconda Python/R Distribution - Anaconda,“ [Online]. Available: https://www.anaconda.com/distribution/, last accessed on 2019-07-05.

[46] „About Debian,“ [Online]. Available: https://www.debian.org/intro/about, last accessed on 2019-07-05.

[47] „About - OpenCV,“ [Online]. Available: https://opencv.org/about/, last accessed on 2019-07-05.

[48] „RAVPower 24 W 4.8 A Dual USB Mini Car Charger,“ [Online]. Available: https://www.amazon.de/Ladeger%C3%A4t-RAVPower-Ladeadapter-Technologie-Powerbank/dp/B01B14AGFM/ref=sr_1_6?keywords=autoladeger%C3%A4t&qid=1562447596&s=gateway&sr=8-6, last accessed on 2019-07-05.