Contents

[Specifications 3](#_Toc14870707)

[Supported modules: 3](#_Toc14870708)

[Libraries used 4](#_Toc14870709)

[Hardware and Raspberry Pi 5](#_Toc14870719)

[Camera Port 5](#_Toc14870720)

[USB Ports 5](#_Toc14870721)

[GPIO Pins 5](#_Toc14870722)

[Installing the Operating System 7](#_Toc14870723)

[Parts Needed 7](#_Toc14870724)

[Installing the ISO 7](#_Toc14870725)

[Installing the operating system 7](#_Toc14870726)

[Updating the operating system 7](#_Toc14870727)

[Logging in 8](#_Toc14870728)

[Accessing the Raspberry Pi remotely through another computer 8](#_Toc14870729)

[Setting up Python and Git 9](#_Toc14870730)

[Verify installation 9](#_Toc14870731)

[Set up a Git repository 9](#_Toc14870732)

[Retrieving changes from a Git repository 9](#_Toc14870733)

[Installing python dependencies 10](#_Toc14870734)

[Installing OpenCV 10](#_Toc14870735)

[Running the program 11](#_Toc14870736)

[Program overview 13](#_Toc14870737)

[File Structure Overview 13](#_Toc14870738)

[Robot Control 14](#_Toc14870739)

# Specifications

* Microcomputer – Raspberry Pi 3B+ X2
  + 32 GB MicroSD Card
  + 4000mAh battery pack for raspberry pi
  + Pololu dual G2 High-Power Motor Driver 18V18 for Raspberry Pi
* FLIRC USB
  + Remote control
* Chassis/Motors – Dagu Wild Thumper 4WD All-Terrain Chassis, Black, 75:1
* 7.2 V battery pack (rechargeable)
* 7.2 V battery pack charger
* Raspberry Pi supported 1080 camera
* Tamiya cable with male connector and stripped cable end
* Breadboard
* Miscellaneous screws and standoffs

## Supported modules:

* External Raspberry Pi 3B+
  + External fisheye camera
* Adafruit ultimate GPS Breakout
  + TTL to USB cable

# Libraries used

# six==1.12.0 – Cross-version compatibility between

# pygame==1.9.6 – Application library used for visual applications as well as remote input

# numpy==1.16.4 – Library which implements arrays into python. Also contains other mathematical functions. Required by opencv\_python

# matplotlib==3.1.0 – Library used to display data. Used by opencv\_python.

# opencv\_python==4.1.0.25 – Powerful visual processing library used for navigation and analysis of images.

# GPSPhoto==2.2.3 – Library that attaches EXIF metadata to images, specifically GPS data.

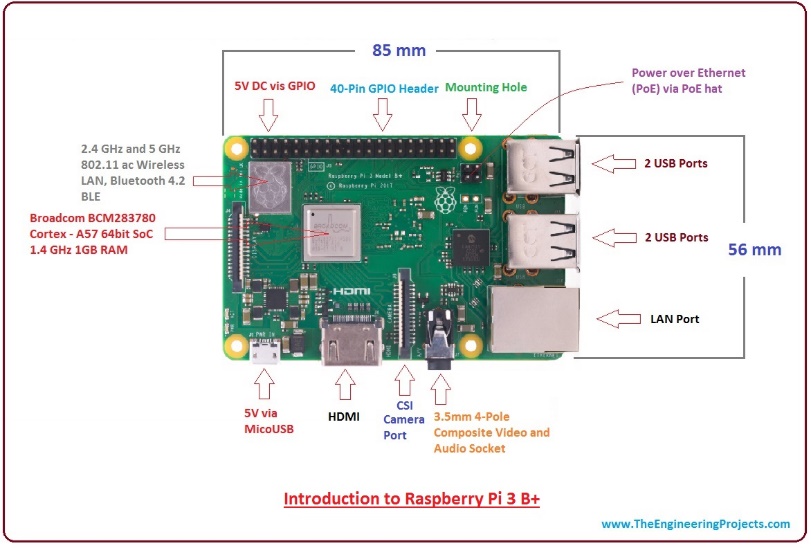
# picamera==1.13 – Library included by default in Raspbian. Allows control of the built in camera slot.

# piexif==1.1.3 – Allows manipulation of image EXIF metadata. Required by GPSPhoto

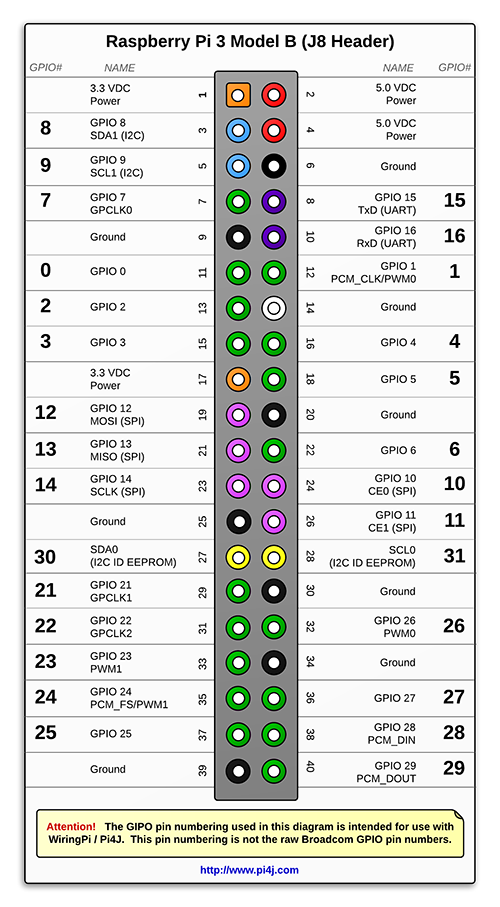
dual\_g2\_hdmp\_rpi – Library provided by Pololu to control their motor controller hat.

# wiringpi==2.46.0 – Used to read GPIO data from the raspberry pi. Used by dual\_g2\_hdmp\_rpi

# Hardware and Raspberry Pi

The raspberry pi is a microcomputer that is used in this project as the logical controller. The raspberry pi is equivalent to a normal computer in that it has its own operating system, multithreading, and a motherboard. In figure 1, you can see the various inputs and outputs of the raspberry pi. Note that the raspberry pi has built in Bluetooth, and wireless capabilities.

## Camera Port

The raspberry pi has a dedicated camera port slot. This allows for high speed transfer of data that a normal USB port cannot achieve. Most attachments to the top of the raspberry pi (called hats) has a slot to pass the camera cord through to allow a camera to be used with the module.

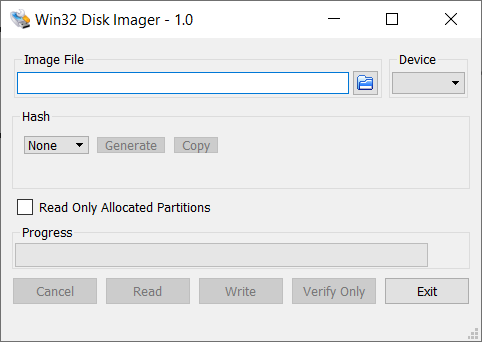
## USB Ports

The raspberry pi comes with four USB type A ports for data transfer. Additionally, the raspberry pi comes with a single 5V micro USB port to power the raspberry pi. Note, the batteries used in this robot supply roughly 3.5 amps. The raspberry pi takes 1 amp to run, not including the power drain through USB ports. While the current sensors used through USB are compatible with the amperage limits, if more amperage than what is supplied to the raspberry pi is used, an externally powered USB hub would be required

## GPIO Pins

See figure 2 for the pinout diagram. The GPIO pins send raw data through multiple different types of ports and configurations. The simplest way to use the GPIO pins for testing is with a breadboard. This allows for wires to be inserted into the breadboard for easy testing without soldering. Note that the I2C ports can be parallel ports, this means that multiple devices can use the I2C ports. All other ports can only be accessed by one device at a time.

# Installing the Operating System

Because the SD card in the raspberry pi may need to be replaced, you may need to reinstall the operating system back onto the raspberry pi.

## Parts Needed

1. Raspberry Pi
2. MicroSD card (At least 8gb)
3. MicroSD to USB converter
4. External computer
5. Mouse
6. Keyboard
7. Monitor
8. HDMI cord

## Installing the ISO

1. Use the preformatted version of Raspbian. Select the desktop version with recommended software
2. Download and install Win32DiskImager [here](https://sourceforge.net/projects/win32diskimager/). Launch the program. It should look like figure 3
3. Select the ISO file you downloaded
4. Keeping all the default settings, press write
5. The program will display a warning message, saying the Micro-SD card will be wiped, press continue
6. Allow the program to mount the ISO onto the Micro-SD card.

## Installing the operating system

1. Insert the micro SD card into the micro SD slot on the back of the raspberry pi
2. Connect a power source to the raspberry pi, it should boot up after a few minutes to the main screen.

## Updating the operating system

In order to update the operating system (which should be done on first startup):

1. Run:

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

## Logging in

1. The default login credentials are:
2. Username: pi
3. Password: Password01

## Accessing the Raspberry Pi remotely through another computer

While connecting with a mouse keyboard and monitor works, a simple and portable solution would be to login using VNC viewer

1. On the console run:

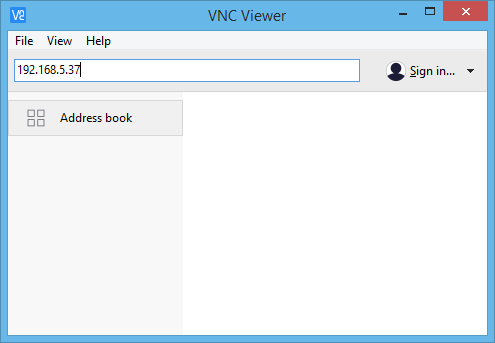
sudo apt-get update  
sudo apt-get install realvnc-vnc-server

IP Address:

sudo raspi-config

1. Navigate to interfacing options > vnc and ensure it is enabled
2. On the raspberry pi console run:

sudo ip addr show

1. Write down this ip address for later use
2. On the remote computer (Windows):  
   [Download VNC viewer](https://www.realvnc.com/en/connect/download/viewer/)  
   Input the ipaddress into the search bar and press enter to establish a connection

# Setting up Python and Git

Note: Both python 2 and python 3 are installed by default on Raspbian. For this project we are using python 3

## Verify installation

1. Open a console window  
   Type these commands to ensure installation:

python3 -–version  
pip –-version  
git

1. These three should return information. If an error message is received, they must be installed

## Set up a Git repository

In order to setup a place to download the program and libraries, a git repository must be set up.

1. Using ls (to show files) and cd (to change directory, or folder) in the console window, navigate to the desktop
2. Use:

mkdir git

To create a folder to store all github projects. Navigate into this folder

1. Use:

sudo git clone https://github.com/

Where github.com is the link to the git project available for export on the project’s website. The link should be: <https://github.com/danielsmith1313/Robot.git>

1. The entire library should now be downloaded.

## Retrieving changes from a Git repository

While we have the code now, it is much easier to edit the code externally and pull the changes

1. Navigate to the project directory under the git directory
2. Type:

sudo git pull

1. The git repository should be updated

## Installing python dependencies

1. Navigate to the requirements.txt folder under the git project
2. For each library in this file install:

sudo pip3 install libraryname

## Installing OpenCV

Note: OpenCV is preinstalled on the raspbian ISO.

# Running the program

1. Navigate to your git folder
2. Navigate to the bin folder
3. In the bin folder, run:

sudo python3 \_\_main\_\_.py

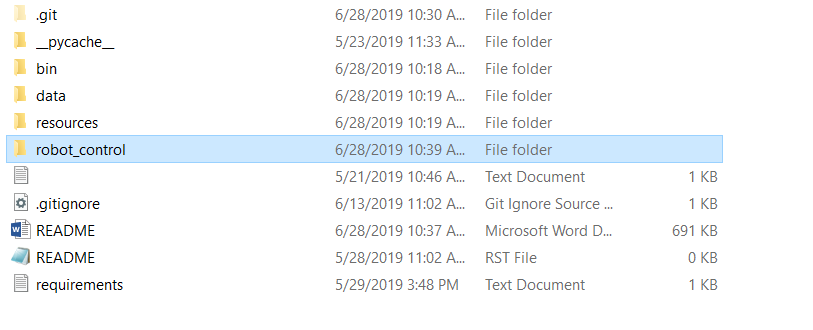
Below is the controller diagram which runs different functions:



|  |  |
| --- | --- |
| STOP/MODE | Stops the execution of the current program |
| Back button | Restarts the raspberry pi |
| SETUP | Powers off the raspberry pi |
| D-Pad | Controls movement of the raspberry pi manually |
| 0/10+ | Changes input mode for the number pad for programmed |

|  |  |
| --- | --- |
| 1 | Takes a picture with the upward facing camera |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |

# Program overview



## File Structure Overview

The main folder of this project contains five main folders. The bin folder (short for binary) contains control\_application gpsexec and \_\_main\_\_.py. The \_\_main\_\_.py folder is reserved for executable programs, programs that are meant to be run directly from the operating system. \_\_main\_\_ is the main program and allows for many functions controllable by an infra-red remote. The gpsexec.py file takes the average of 100 GPS readings and prints the result. The control\_application.py file allows for control of the robot through a computer user interface.

The data folder in this project contains all the data gathered by the robot. It is a place to store readings, images and any other kind of data cameras or sensors.

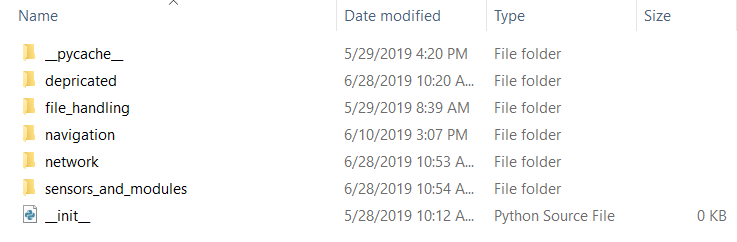
The resources folder contains any non-code files that are used in the program

The robot\_control folder contains reusable code.

The two README files are this current file, one as a microsoft word document and another as a ReStructured text file. All documentation should be done on the Microsoft word document, the ReStructured text file is for online documentation in github.

The requirements.txt is an auto-generated file which contains all libraries required for python as well as the versions.

## Robot Control



The robot\_control folder contains reusable code used in the main program.

The deprecated folder contains programs that have already been replaced or are in the process of being replaced.

The folder file\_handling contains a class which converts python data such as a list integer, etc. into writeable text.

The folder navigation contains the class which allows the robot to navigate through a field and take sensor data

Network contains a script that sends a signal to run a certain file on another raspberry pi

Sensors\_and\_modules contains distance\_sensor.py which is used to read the distance sensor on the robot, camera.py takes a picture and attaches GPS EXIF data to it. This script should be compatible with any camera attached to the dedicated camera slot. The file gps\_reader.py reads the GPS data from the sensor.