Remote-controlled Camera Robot
Table of Contents

LIST OF FIGURES AND TABLES ........................................................................................................... 4

Figures .................................................................................................................................................. 4
Tables .................................................................................................................................................. 4

SUMMARY ............................................................................................................................................... 5

INTRODUCTION ..................................................................................................................................... 6

PROJECT OVERVIEW ............................................................................................................................ 7

Equipment ............................................................................................................................................ 7
Procedure ............................................................................................................................................... 7
COSTS ................................................................................................................................................... 8
SCHEDULE ........................................................................................................................................... 8
Block Diagram ...................................................................................................................................... 9

HARDWARE DETAILS AND INSTALLATION ....................................................................................... 9

Raspberry Pi .......................................................................................................................................... 9
RPI Camera .......................................................................................................................................... 11
Wheels and Motors .............................................................................................................................. 11
What is a motor controller “H-Bridge" and why do we need it? ......................................................... 12

Features .............................................................................................................................................. 12
How to use .......................................................................................................................................... 13

Motors and wheels Installation ............................................................................................................. 13

Step 1: Installing the extra DC Power Supply for Motors ................................................................. 13
Step 2: Connector Wires for Driving the Direction on the Motor: ................................................. 13

Troubleshooting ................................................................................................................................ 14

Hardware Problems That We Faced .................................................................................................. 14
Why Did Our Raspberry Pi Burn?????? ............................................................................................. 14
Cellular Data Dongle ............................................................................................................................ 15

SOFTWARE .......................................................................................................................................... 16

The Server ........................................................................................................................................... 16
RPI Cam Web Interface ......................................................................................................................... 16
LIST OF FIGURES AND TABLES

Figures
Figure 1: Parts (citations from left to right: [1], [2], [3]) ................................................................. 7
Figure 2: Block Diagram .......................................................................................................................... 9
Figure 3: RPi Camera Module ............................................................................................................... 11
Figure 4: Waveshare onboard L293D Dual H-Bridge Motor Control Driver Module ......................... 12
Figure 5: H-Bridge Wiring Diagram ...................................................................................................... 13
Figure 6: Breadboard for testing wire connections .............................................................................. 14
Figure 7: RPi Cam Display .................................................................................................................... 17
Figure 8: PiTunnel.com .......................................................................................................................... 19
Figure 9: Downloading NOOBS ........................................................................................................... 22
Figure 10: Raspberry Pi Pinout ........................................................................................................... 24

Tables
Table 1: Original Costs .......................................................................................................................... 8
Table 2: Time Schedule .......................................................................................................................... 8
Table 3: Hardware Components and Specifications for Raspberry Pi 2B ............................................. 10
Table 4: I/O Diagram for H-Bridge ....................................................................................................... 13
SUMMARY

This report is the outcome of our capstone project named “Remote-Controlled Camera Robot”. This project was completed by students from BCIT’s Telecommunications option in the ECET department. As a completed project, we successfully built a robot car which can be controlled from a web interface through WiFi or a data cellular system. Using a webpage, the user can access commands that allow the robot to move forward, back, and turn. The video feed from the robot is also visible here, and the user can send simple commands to the camera. For example, we can record video, turn on/off the camera, get picture capture, etc. This project is not only successful according to the technical point, but has also been completed within the limited budget and timeline.

We broke down the project into two parts, which are hardware and software. The hardware part includes a 4-wheel chassis, Raspberry Pi, camera module, WiFi dongle, etc. In the software part, we have used Apache 2, HTML, Java, PHP, PiTunnel, etc. After building the hardware and software part separately, we can now control the robot car with Raspberry Pi, and communicate with it through a webpage.

We have created a very basic robot car which can controlled by a webpage but there are lots of other possibilities to make it more efficient. Such as stronger motors/wheels, a better camera, improvement of the power supply or batteries. These future improvements will be explained at the end of the document in the FUTURE RECOMMENDATIONS section.
INTRODUCTION

Humans are not indestructible and sometimes it is better to send a robot in its place. For example, this can happen due to natural disasters or military zones where land mines are present. We want to be able to safely scout out these areas for information or even survivors.

To address these problems, we created a small robot, able to fit in small places and withstand the elements that humans normally cannot. It will be complete with wheels and a camera, all of which can be controlled remotely via Wi-Fi. We also created a webpage that allows a user to control the multi-directional movements and receive video feed from the robot.

BCIT’s ECET department always encourages students to do a capstone project in their last term according to the students’ interests, or instead, a project from a sponsor. As students of the telecommunication and network option, our group took this project because of our own interest in wireless communication, robotics and data processing. The departmental head of the telecommunication and network option, Ed Casas, and the project supervisor, Robert Trost, closely monitored and guided us to make the project working throughout the term. Our communication instructor, Susan Woo, supervised us to make the documentation for this project.

This documentation will provide enough information to rebuild this project or make any improvement in future. For the hardware part of this project, we have used equipment such as a Raspberry Pi 2B, camera module and 4-wheel chassis. We found lots of open source software such as HTML, Java Script, Apache 2, PHP, and PiTunnel.

We have displayed our budget from BCIT for this project, in Table 1, and our time management, in Table 2. In our block diagram of the project, shown in Figure 2: Block Diagram, we can see how the wheels, camera and Raspberry Pi get connected with the webpage.
PROJECT OVERVIEW
The project is mainly focused on programming the Raspberry pi and making the webpage. We also needed to install the car with wheels, motors and some other hardware parts. After we organize the project, we divided the total work into two sections, hardware, and software.

Equipment
Most of this project is made up of programming, but some hardware equipment is also very important to make the full project working. The parts list is listed below, along with Figure 1 showing some sample pictures.

- Monitor, mouse, keyboard
- HDMI cable
- Raspberry pi 2B
- Camera (for raspberry pi)
- Chassis with four DC motors and wheels
- Two Battery Packs: one for powering up the Raspberry Pi and one for driving the motors which is a pack of six AA batteries
- L293D Motor Control Board (H-bridge)
- Wires
- Soldering iron
- Ethernet cable, Wi-Fi dongle, and data connection dongle

Procedure
1) Get parts
2) Download the required programs onto the raspberry pi and install Python
3) Program commands for wheels to move forwards, backwards, and turn (test with LEDs)
4) Create a web page
5) Program controls on the web page to make the wheels move forwards, backwards, turn
6) Connect the camera to the Raspberry Pi
7) Make controls for the camera on the web page
8) Record video from the pi and send it to the web page

Figure 1: Parts (citations from left to right: [1], [2], [3])
COSTS

After searching various web sites provided by our instructor, we can up with a budget plan. See Table 1 for details on the costs. This changed however, because we found almost all of our supplies from other instructors. We ended up using a Raspberry Pi model 2B instead of the 3B, but we didn’t have to spend any money on it. Our only purchase was the 4-wheel chassis combo from Lee’s Electronics. [2]

Table 1: Original Costs

<table>
<thead>
<tr>
<th>Description</th>
<th>Price per unit</th>
<th>Units</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raspberry Pi 3b</td>
<td>$55.00</td>
<td>1</td>
<td>$55.00</td>
</tr>
<tr>
<td>Camera with cable</td>
<td>$30.00</td>
<td>1</td>
<td>$30.00</td>
</tr>
<tr>
<td>Wheels &amp; Motors</td>
<td>$25.00</td>
<td>1</td>
<td>$20.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>$105.00</strong></td>
</tr>
</tbody>
</table>

Our budget limitation for this project was $200, which we stayed well under at only $25 for the 4-wheel chassis.

SCHEDULE

This project’s final due date was November 27. Therefore, we created a schedule to fit that time frame (see Table 2). All three of us put work into each area as a group in our project labs. The only adjustment we made to our original schedule was delaying the interfacing of the wheels by seven days.

Table 2: Time Schedule

<table>
<thead>
<tr>
<th>Due Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep 29</td>
<td>Get parts</td>
</tr>
<tr>
<td>Oct 3</td>
<td>Software setup</td>
</tr>
<tr>
<td>Oct 15</td>
<td>Program wheels</td>
</tr>
<tr>
<td>Oct 26</td>
<td>Create web page</td>
</tr>
<tr>
<td>Oct 30</td>
<td>Interface wheels with web page</td>
</tr>
<tr>
<td>Oct 30</td>
<td>Add turning</td>
</tr>
<tr>
<td>Nov 9</td>
<td>Connect and program camera</td>
</tr>
<tr>
<td>Nov 23</td>
<td>Include video streaming function to the web page</td>
</tr>
<tr>
<td>Nov 26</td>
<td>Final testing</td>
</tr>
</tbody>
</table>
Block Diagram
We are talking about a robot which has motion in 4 directions and are controllable from a distance. For these features, we are using a Raspberry Pi as our central controller to be able to get the user commands through the web interface and send the proper commands to the motors which are controlling the wheels. The overview of the project, shown in Figure 2: Block Diagram, illustrate how hardware and software get connected with each other.

![Block Diagram](image)

HARDWARE DETAILS AND INSTALLATION
This section will provide details on the hardware specifications that was used, and also the installation process.

Raspberry Pi
The first step of installing our hardware and starting the project is setting up the Raspberry Pi. We chose Raspberry Pi as our controller because it is a mini computer that is specifically created to make tech learning easier. It is cheap and easy to upgrade and add features to the project later.
It has a lot of components for computer-based projects, like USB ports, an Ethernet port, an SD card slot, Wi-Fi antenna ports, and more. Below, in Table 3: Hardware Components and Specifications for Raspberry Pi 2B are some details of the model that we are using, Raspberry Pi 2B.

Table 3: Hardware Components and Specifications for Raspberry Pi 2B

<table>
<thead>
<tr>
<th>CPU and Processor</th>
<th>Quad-core ARM Cortex-A7 CPU 0.9 GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>1GB SDRAM</td>
</tr>
<tr>
<td>Ports</td>
<td>Audio: 3.5 mm jack, Storage: microSD card slot, Network: 10/100 Mb/s</td>
</tr>
<tr>
<td>Power</td>
<td>Micro USB DC 5 V 800 mA for the Pi board</td>
</tr>
</tbody>
</table>

There are a few components you will need before you can get started for setting up and coding the Raspberry Pi:

- Power supply for Raspberry Pi you can easily use the Android cellphone charger or lab’s power supply. But test the current to be in range to prevent the damage the raspberry
- USB keyboard
- USB mouse
- An HDMI cables
- An ethernet cable (or Wi-Fi dongle)

A connection to the Internet is not required for setup, but many Raspberry Pi projects use them.

- A monitor
- microSD card

The microSD card must have at least 8 GB of storage. You can purchase one that comes pre-loaded with Raspberry Pi’s New Out of Box Software (NOOBS), but you can also download the software for free from the website, so there is no need to purchase a special NOOBS microSD card. [4] See APPENDIX A: RASPBERRY PI INSTALLATION for more information on how to set up the Raspberry from scratch.
RPi Camera

The Camera module that we used, shown in Figure 3: RPi Camera Module, is Raspberry Pi Camera Module V2-8 Megapixel, 1080p. This section will explain how to install the Raspberry Pi camera module:

![Figure 3: RPi Camera Module](image)

1. This module is sensitive to static electricity. So, you need to discharge yourself before removing the camera from its cover.
2. Insert the Camera module’s cable into the Raspberry Pi. The cable slots into the connector situated between the Ethernet and HDMI ports, with the silver connectors facing the HDMI port.
3. Reboot the Raspberry Pi.
4. Run "sudo raspi-config" in command window. If the "camera" option is not listed, you will need to update your Raspberry Pi: Run "sudo apt-get update" and "sudo apt-get upgrade" to update it.
5. To check if camera module is identified by Raspberry, Run "sudo raspi-config" again. The "camera" option should be there now.
6. It is necessary to go to the configuration menu and enable the camera.
7. Reboot the Raspberry Pi

Now it is ready to program.

Wheels and Motors

In this project we are using 4PCs DC Electric Motor 3-6V with 4Pcs Plastic Toy Car Tire Wheel, Mini Smart RC Car Robot Tires.

We connected two motors on one side together, and the same on the other side. In this case all we need is two commands for the robot’s motions, meaning that the robot uses 2 pairs of powered wheels. Each pair turns in the same direction. For going forward, we send the forward command to both sides. And the backward command to both sides for going backward. But
what about turning? We are using Point Turn Concepts. For turning, the car’s one side must go forward while the other side goes backwards.

But we can’t connect the motors directly to the Raspberry Pi pins. We need a motor controller board.

**What is a motor controller “H-Bridge" and why do we need it?**

When we connect the + wire of battery pack on one lead of DC motor and - wire on the other lead, the motor turns. Now if we switch the polarity connections, the motor turns the other direction. For switching the electric polarity, we need H-Bridge board.

In this Project we are using a Waveshare Onboard L293D Dual H-Bridge, shown in Figure 4: Waveshare onboard L293D Dual H-Bridge Motor Control Driver Module. [3] This means we have the ability to control two motors in either direction. Table 4: I/O Diagram for H-Bridge shows the input and output characteristics of the H-bridge.

![Figure 4: Waveshare onboard L293D Dual H-Bridge Motor Control Driver Module](image)

**Features**

- Onboard dual H-bridge driver L293D
  - Drives 2 DC motors or 1 stepping motor at one time
  - 600mA output current (peak 1.2A) per single bridge
  - ESD protection
- 5V operating voltage, motor drive voltage 4.5V-36V
  - Output via screw terminals, easy connection
- Dimension: 43mm*27mm
- Mounting hole size: 2mm
- Storage temperature: -25°C ~ +130°C
How to use

- M1A/M1B: connects to motors in right side rotation controlling
- M2A/M2: connects to motors in left side rotation controlling
- GND: ground
- VCC: 5V power supply

Table 4: I/O Diagram for H-Bridge

<table>
<thead>
<tr>
<th>M1A/M2A</th>
<th>M1B/M2B</th>
<th>M1/M2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>Counterclockwise/Clockwise</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Clockwise/Counterclockwise</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Stop</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Stop</td>
</tr>
</tbody>
</table>

Motors and wheels Installation

After gathering the parts, we now needed to solder wires to the motors, and connect them to the H-bridge, and Raspberry Pi.

Step 1: Installing the extra DC Power Supply for Motors

In this Project we are using two power supplies, one for the Raspberry Pi and other one for motors. Each motor needs a DC power supply in the Voltage range of 3-6V. We used the six AA battery Pack to drive our motors.

We learned to NEVER USE RASPBERRY PI’s 5V source to drive the motors. We tested them this way which resulted in a burnt Raspberry Pi board!!!!

Step 2: Connector Wires for Driving the Direction on the Motor:

Figure 5: H-Bridge Wiring Diagram shows a wiring diagram of the H-bridge connected to each motor.
The "GPIO" pins on the Raspberry Pi will be used to trigger the L293D direction.

We needed to attach a jumper wire from the + and MP pins on the H-bridge. This connected our external power supply to our motors. Since the WiringPi pin numbers of the Raspberry Pi are different versus the physical pin numbering we should be careful about wiring the wrong pins. These can be seen in APPENDIX B: RASPBERRY PI PIN. It happened that our code was not controlling the motors on/off directions, we figure out that our physical wires connected to the wrong pins different from GPIO in our code.

Troubleshooting
We checked our physical pins, which we expected to be our motor controller inputs, with LEDs to make sure they our code was working properly. Our test setup is shown in Figure 6: Breadboard for testing wire connections.

Hardware Problems That We Faced
We faced a number of problems, including burning our Raspberry Pi and connecting to WiFi.

Why Did Our Raspberry Pi Burn?????
Our Raspberry Pi burn as a result of drawing too much current from its GPIO pins. The mistake was connecting the motor pins directly to raspberry pins. Since motors draw more current in compare with LEDs, it is necessary to have separate power supply for motors.
Note that the maximum recommended current draw from the 3.3 V power output pins is 50 mA that is not enough for driving motors.

**Cellular Data Dongle**

We want our car to be able to go anywhere and still be able to control it from distance, so we needed internet connection. Since WiFi is not available everywhere, our next challenge was using cellular connectivity and data SIM. We used Rogers’s network mobile SIM card. The broadband dongle plugs straight into the Raspberry Pi USB port, and allows it to access the internet on the go. We used the following link to help us install the cellular data package: [https://blog.soracom.io/beginners-guide-to-iot-cellular-connectivity-on-raspberry-pi-and-linux-devices-55d4f7489adf](https://blog.soracom.io/beginners-guide-to-iot-cellular-connectivity-on-raspberry-pi-and-linux-devices-55d4f7489adf) [5]

![Cellular Data Dongle](image)

**Installation of the Data Dongle:**

1. First we installed the network-manager package:

   ```
   pi@raspberrypi:~ $ sudo apt-get update && sudo apt-get install network-manager
   ```

2. After installing the network manager successfully, we made the 3G Dongle USB connections. We needed to add Rogers’ network Access Point Name (APN):

   ```
   pi@raspberrypi:~ $ sudo nmcli con add type gsm ifname "*" con-name rogers apn itemobile.apn
   ```

3. Reboot the Pi

   ```
   pi@raspberrypi:~ $ sudo reboot
   ```

4. Once the Pi started rebooting, we hot plugged the USB Dongle to the Pi

   The Pi is now connected to Rogers’ cellular network. Each time a Dongle is plugged in, the Pi is restarted, or connection drops, it will automatically reconnect to the network.
SOFTWARE

After installing Raspbian, we could now add the rest of the software components. While testing the motion for the wheels, we attempted to use Python and Computer Graphics Interface (CGI). This worked initially, however we were unable to make a connection to our server this way. The following, outlines the software that we used in our final product.

The Server

Our original idea was to use a free web hosting server online. Upon experimenting with this, we realized the server required the IP address of the Raspberry Pi in order to communicate with it. This proved to be difficult as the Pi would potentially have a different IP address each time it connects to the internet.

The easy solution was to instead create the server on the Pi itself. We chose apache as our server because it works well with Raspbian and there is lots of helpful documentation online as well. Installation instructions for apache 2 can be found in APPENDIX C: SOFTWARE INSTALLATION

The server can be accessed by typing the local IP address of the Pi into a web browser. At first, we were only able to access the server if we were in the same local area network as the Pi. This issue will be discussed later in the PiTunnel section of this document. It should also be noted that we used JavaScript for our project, so it will need to be enabled on the browser that is being used.

RPi Cam Web Interface

There is an excellent, open-source interface for the Raspberry Pi camera module that can be downloaded from: https://github.com/silvanmelchior/RPi_Cam_Web_Interface [6]

Documentation for this software can be found at: https://elinux.org/RPi-Cam-Web-Interface [7]

After extracting the zip file, the folder needed to be moved into the Raspberry Pi directory: /var/www. Our apache server is set up to run code in this directory by default. Further installation instructions for this software can be found on the website [7] or in APPENDIX C: SOFTWARE INSTALLATION of this document. We will also need to install the WiringPi library, which contains commands for controlling the Pi’s GPIO pins. Instructions for this can also be found in APPENDIX C: SOFTWARE INSTALLATION.

This package contains code for a variety of camera features such as low-latency video streaming, recording, motion detection, and even system controls for the Pi, such as rebooting the device. Figure 7: RPi Cam Display displays what will be shown on a web browser whenever the server is accessed from a computer or any other device with internet capabilities.
Figure 7: RPi Cam Display

This interface only works with apache, nginx, or lighttpd servers. Additionally, the program is only designed for a Raspberry Pi camera.

Raspberry Pi Permissions
Most files created on Raspberry Pi will only be readable by default. In order to write to and/or execute a file, use the following command:

➢ chmod -R 777 filename

Any additional permissions issues will most likely be solved by typing sudo right before the desired command.

Wheels Interface
The code for the RPi Cam Web Interface was written using HTML, JavaScript, and PHP. This influenced us to program our interface for the wheels using these languages as well. Instead of creating new files for our program, it was easier to add our code to the existing files in RPi Cam. Three files were edited in this way and the code added can be seen in APPENDIX D: CODE FOR THE WHEELS.
**Index.php** contains HTML code to create the visual layout that can be seen in Figure 7: RPi Cam Display. We also added a line of text that says: 'Press w,a,s,d to move. To pivot left, press q, pivot right, press e.' This file also calls the functions keyUp() and getKeyPress(), which are defined in the script.js file.

**Script.js** is a JavaScript file that will contain functions related to user interactions on the web page such as clicking a button. The getKeyPress() function that we added will detect when a user presses a specific key. The keyUp() function detects when a user releases a key. Both functions will set a variable called ‘cmd’ to the corresponding command we want. For example, if the user presses ‘w’, cmd will be set to ‘forward’. We then use the AJAX GET method to send the cmd variable to cmd_func.php.

AJAX (Asynchronous JavaScript And XML) is a useful set of functions that were created to help with web development. It allows us to collect, update, and send data between our web page and server without having to refresh the page. The GET method is one example of a function we can use to send data from our web page to our web server. [8]

**Cmd_func.php** contains the code for the various commands we can run, such as moving the car forward. Php is a server-side language, so it allows us to control aspects of the Raspberry Pi (because our server is set up on the Pi itself). As an example, if cmd = ‘forward’, then cmd_func.php will turn on the corresponding Raspberry Pi GPIO pins to make the car move forward. It should be noted that the pins we use for our code are the WiringPi library’s pins. These are different from the Pi’s physical pins. A pinout diagram can be seen in APPENDIX B: RASPBERRY PI PINOUT

**PiTunnel**
As mentioned earlier, our web server was only accessible from devices that were on the same local area network. The most common solution to this would be to configure port forwarding on the router being used. As we did not have access to the router, we needed an alternative. We found a very useful website to assist us with this issue at: [https://pitunnel.com](https://pitunnel.com). [9]

PiTunnel uses tunneling, which is a protocol that allows data of private networks to be sent over a public network. This method also gives our data packets encryption, making our data very secure. A free account on PiTunnel provides 100Mb/day of data transfer, which is perfect for a short demonstration. For long-term use, we would need to pay for a monthly subscription.

Figure 8: PiTunnel.com shows an example of what the website looks like.
To initiate communication to PiTunnel, we followed the installation instructions on the website. These commands can also be found in APPENDIX C: SOFTWARE INSTALLATION of this document. Now, whenever the Pi starts up with internet connection, PiTunnel will create a URL link in the ‘Active Tunnels’ section of PiTunnel.com. For example, in Figure 8: PiTunnel.com, the user may click on pitunnel.com:52123 to bring them to our webpage. The number, 52123, is the port number on PiTunnel that connects to port 80 of the Pi. Each time the Pi connects to the internet, a new port number will be generated.

Another useful feature of PiTunnel is the remote terminal. This provides access to the Pi’s command terminal, without the need of having to connect a mouse, keyboard, and monitor to the Pi.
CONCLUSION

Our goal for this project was to create a robot car which can be controlled by a webpage wirelessly and when receiving any command like moving the car or recording video, it will function properly. Although we had some problems like not getting connection from WiFi, burning the Raspberry Pi, and accessing the server on a public network, we were able to fix all of those problems and make a working project.

FUTURE RECOMMENDATIONS

This project has various options for future improvements, that we have listed below:

- Stronger motors/ wheels
- Powerful Battery / Solar Panel attached
- Improve camera quality
- Bluetooth instead of using cellular data/ WiFi
- Mobile App

As we had a budget limitation of $200, the wheels and motors we have used can only run on a smooth surface. Using stronger motors and wheels, the car could run on rough surfaces or climb hills. The battery pack and power bank we have used can only give power to the car for a few hours, but a more powerful battery or solar PV could make the robot run for a longer period of time.

The Raspberry Pi camera module we used could only view directly in front of the robot, but we can improve the robot’s vision by adding a pivoting structure and a command to turn the camera independently from the robot. When WiFi or Cellular data isn’t available Bluetooth is another option for getting connected with the webpage.

We can control the car with any device like a laptop or mobile device but making a mobile app and for this project would be one of the most important features to add in the future.
REFERENCES


APPENDIX A: RASPBERRY PI INSTALLATION

Most likely the new Raspberry Pi comes with the preinstalled Raspbian operating system on an SD card. If not, the first thing to do is installing it. For this action we need another computer with an SD card reader to install the image.

Step 1: Reformat your microSD card
Since we got the Raspberry from a previous project, it had some extraneous files on them. Reformatting it will remove all files and completely clear the card. So, our first step to getting started with our Raspberry Pi was to reformat the microSD card before downloading the operating system on it. So, we need an SD card adapter to be able to use our laptop card reader (if the laptop doesn’t have card reader it is easy to simply use the USB card reader as well). Now download an SD formatter and format the SD card.

Step 2: Download NOOBS onto the microSD card
Now it is time to get NOOBS onto the microSD card. We downloaded NOOBS from the Raspberry Pi website: https://www.raspberrypi.org/downloads/noobs/. [4] Below, in Figure 9: Downloading NOOBS shows how the NOOBS website looks like.

Once it’s loaded, we could plug it into the Raspberry Pi and configure the operating system. Since we want to use the Raspbian we downloaded the ZIP file of NOOBS Version 3.0.0 and not NOOBS Lite. It is a large file and took a while to complete.

![Figure 9: Downloading NOOBS](https://www.raspberrypi.org/downloads/noobs/)
When the download completes:

1. Open the zipped NOOBs folder
2. Select all files and folders
3. Drag and drop all selected NOOBS files into the SD card.

Now it’s time to plug in the SD card to Raspberry Pi:

1. Right-click on the SD card icon.
2. Select Eject SD Card to remove the card reader from you PC safely.
3. Remove the card reader from your computer.
4. Remove the microSD card from the card reader.

**Step 3: Set up your Raspberry Pi**

1. Plug in the microSD card into the Raspberry Pi card slot.
2. Plug the USB mouse into one of the USB ports
3. Plug the USB keyboard into one of the USB ports
4. Connect the Raspberry Pi to the monitor using the HDMI cable
5. Connect an ethernet cable to be able to use the Internet
6. Power up the Raspberry Pi

**Step 4: Download the Raspbian operating system on the Raspberry Pi**

After Powering on the Raspberry, a start-up screen showed up. We are using the Raspbian operating system:

1. Select Raspbian.
2. Click Install.

The Raspbian installation took a while. Once the installation process is finished, Raspbian automatically began to boot.

**Step 5: Configure your Raspberry Pi**

After installing the Raspbian we need to configure the Raspberry Pi system in order to add the location, date, and time. These windows were pop up to setup, but they are accessible through Menu as well:

1. Click Menu in the upper left corner of the screen.
2. Select Preferences in the dropdown menu.
3. Select Raspberry Pi Configuration under Preferences
4. Set the Location, Time zone, and Keyboard language

Now, the Raspberry Pi is ready to use and Program.
APPENDIX B: RASPBERRY PI PINOUT

The Raspberry Pi’s pinout configuration can be seen in Figure 10: Raspberry Pi Pinout. It is also cross-referenced with the WiringPi library’s pinouts. These pin numbers differ because WiringPi was a library originally created for the Arduino microcontroller.

![Raspberry Pi Pinout Diagram](image)

**Figure 10: Raspberry Pi Pinout**
APPENDIX C: SOFTWARE INSTALLATION

This appendix contains the additional lines of code that are required for installation of the following software. These lines are meant to be directly entered into the Raspberry Pi’s terminal.

Apache 2

➢ sudo apt-get update
➢ sudo apt-get upgrade
➢ sudo apt-get install apache2 -y

PHP

➢ sudo apt-get install php libapache2-mod-php -y

RPi Cam Web Interface

(After downloading the RPi Cam package from GitHub and moving the extracted folder to /var/www)

➢ cd /var/www/RPi_Cam_Web_Interface
➢ ./install.sh
➢ sudo /etc/init.d/apache2 restart

WiringPi

➢ sudo apt-get install git git-core
➢ git clone git://git.drogon.net/wiringPi
➢ cd wiringPi
➢ ./build

PiTunnel

➢ curl https://pitunnel.com/install/JUI0pZcRw | sudo python

This next command can be entered in the command line, but if added to the /etc/rc.local file on the Raspberry Pi, it will execute whenever the Pi starts up.

➢ pitunnel --port=80 --http --quiet &
APPENDIX D: CODE FOR THE WHEELS

Code for the wheels interface was added to existing files of the RPi Cam Web Interface. Shown below is the code we added and which files they were added to.

index.php

This section of code was added after the secondary-buttons section in index.php

```html
<div class="container-fluid text-center"></div>
<div class="container-fluid text-center">
<?php echo('Press w,a,s,d to move. To pivot left, press q, pivot right, press e.');?></div>
</div>
</div>
<div class="container-fluid text-center"></div>
<script type="text/javascript">
keyUp();
getKeyPress();
</script>

script.js

script.js can be found in the js folder. These are just functions so they can be added anywhere in the file.

```javascript
function getKeyPress() {
    ajax_status.open("GET", "cmd_func.php?cmd=reset", true);
    ajax_status.send();

    document.onkeypress = function (e) {
        e = e || window.event;
        k = e.keyCode

        if(k == 119){ //w key is pressed
            ajax_status.open("GET", "cmd_func.php?cmd=forward", true);
        } else if(k == 97){ //a
            ajax_status.open("GET", "cmd_func.php?cmd=left", true);
        } else if(k == 115){ //s
            ajax_status.open("GET", "cmd_func.php?cmd=reverse", true);
        } else if(k == 100){ //d
            ajax_status.open("GET", "cmd_func.php?cmd=right", true);
    ```
```php
<?php

define('BASE_DIR', dirname(__FILE__));
require_once(BASE_DIR.'/config.php');

//important to note, WiringPi library's pin numbers are different from Raspberry Pi's physical pin numbers
system ('gpio mode 4 out'); //pin 16 on Raspberry Pi
system ('gpio mode 0 out'); //pin 11
system ('gpio mode 2 out'); //pin 13
system ('gpio mode 3 out'); //pin 15
```
We must also add the forward, reverse, etc. commands inside the function sys_cmd($cmd). This code can be anywhere underneath it using else if statements.

```c
//direction commands
else if(strcmp($cmd, "reset") == 0) {
    system ('gpio write 4 0');
    system ('gpio write 0 0');
    system ('gpio write 2 0');
    system ('gpio write 3 0');
}
else if(strcmp($cmd, "forward") == 0) {
    system ('gpio write 4 0');
    system ('gpio write 0 1');
    system ('gpio write 2 1');
    system ('gpio write 3 0');
}
else if(strcmp($cmd, "left") == 0) {
    system ('gpio write 4 1');
    system ('gpio write 0 1');
    system ('gpio write 2 1');
    system ('gpio write 3 0');
}
else if(strcmp($cmd, "reverse") == 0) {
    system ('gpio write 4 1');
    system ('gpio write 0 0');
    system ('gpio write 2 0');
    system ('gpio write 3 1');
}
else if(strcmp($cmd, "right") == 0) {
    system ('gpio write 4 0');
    system ('gpio write 0 1');
    system ('gpio write 2 0');
    system ('gpio write 3 0');
}
else if(strcmp($cmd, "pivot_left") == 0) {
    system ('gpio write 4 1');
    system ('gpio write 0 0');
    system ('gpio write 2 1');
    system ('gpio write 3 0');
}
else if(strcmp($cmd, "pivot_right") == 0) {
    system ('gpio write 4 0');
    system ('gpio write 0 1');
    system ('gpio write 2 0');
    system ('gpio write 3 1');
}```