



# Project Report

## Program: ESE 4009\_2

**INSTRUCTORS:** Prof. Takis Zourntos, Prof. Mike Aleshams

Group# 10

Student Name	Student ID	Signature*
Ryaan Verma	C0-715940	R.V
Sahib Aden	C0-721078	S.A

*\*By signing above you attest that you have contributed to this submission and confirm that all work you have contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties.*

# **Bike Safety Gear**

## **Abstract**

This report provides a detailed insight into the team's final semester project. The aim of this report is to bring the our project's final form to justification, and to explain the requirement and justify the necessity of such a compact and mobile feedback system for bike riders.

Also, the report talks about the team's approach to such a project, its ideology behind it, and its strategy to deal with the countless problems and barriers faced during the development and final implementation cycle.

Main motto of our team since the beginning of our final semester was to invent/develop a project which would be helpful to a specific section of a society. Considering the self-funded nature of this project, the team took time to observe daily life scenarios of people, and started to list things which could benefit with embedded electronics infused in those scenarios.

Considering the self-funded nature of our project, and after thorough consideration of our personal list of various scenarios experienced by society, we finally selected bikers as our subject, and started brainstorming on what was to become the Bike Gear Assistant.

Collecting the data from various internet sites, and personal interviews conducted about accidents related to bikers in Toronto Downtown, our vision of the project solidified.

We hope to transform our project into a commercial product so as to help decrease the fatality and accident rate of bikers and civilians in general.

# Table of Contents

<b>Acknowledgments</b>	<b>5</b>
<b>Introduction</b>	<b>6</b>
<b>Rationale</b>	<b>6</b>
<b>Hardware Requirements</b>	<b>7</b>
<b>OS and Software Requirements</b>	<b>7</b>
<b>Datasheets and Schematics</b>	<b>8</b>
Arduino Uno R3	8
Details	8
Pin Diagram	9
HC - SR04 Ultrasonic Sensor	10
Details	10
Timing Diagram	10
Raspberry Pi 3 B+	11
Details	11
Kuman 3.5 Inches Display	12
Features	12
Specifications	12
Pi Camera	13
Features	13
Specifications	13
<b>Strategy</b>	<b>14</b>
1. Ultrasonic Sensor Implementation	14
2. Vibrational Feedback Implementation	15
3. Video Feedback using Pi Camera and Pi Zero	15
4. Implementation of virtual Heads-up Display	16

<b>Working</b>	<b>16</b>
1. Ultrasonic sensors with Arduino	17
Interfacing-	17
Initialization-	17
Utilization of Collected Data-	18
2. Vibration Feedback System	19
Interfacing	19
Operation	19
3. LCD Display and Pi camera with Raspberry Pi	20
Interfacing LCD	20
Configuring and Enabling SPI Interfacing	21
Installing WiringPi	22
Installing LCD drivers	22
Interfacing Pi Camera	24
Working of Pi Camera and Display with Raspberry Pi	26
Problems with this approach	26
Solution	27
4. Heads-up Display Mechanism	29
Setup and Working	29
<b>Block Diagram</b>	<b>31</b>
<b>Schematic Diagrams</b>	<b>32</b>
<b>PCB Design</b>	<b>34</b>
<b>Software Architecture</b>	<b>35</b>
<b>Results</b>	<b>36</b>
<b>Conclusion</b>	<b>36</b>
<b>References</b>	<b>37</b>
<b>Appendix</b>	<b>38</b>

## **Acknowledgments**

We would like to thank Prof. Takis Zourntos for his selfless support and effective brainstorming techniques which helped us to conceive the idea of our project.

We'd also like to thank Prof. Mike Aleshams for lending his expertise in documentation, project management and inspirational support.

## Introduction

This project which is the team's inception, is related to the safety of bikers and motorcyclists alike.

The Bike Gear Assistant is a helmet comprising of sensory and visual feedback of the traffic outside of the biker's peripheral vision.

It consists of 4 different systems working in conjunction to one another:

1. *Ultrasonic distance detection system*- to detect any oncoming traffic and send the data to the processor.
2. *Haptic feedback system*- to give the vibrational feedback pulses inversely proportional to the distance detected by the ultrasonic detection system.
3. *Heads-Up display*- to give the real-time video of the rear-view projected in 3D space using virtual-image optics.
4. *Real-time camera feed*- to capture and provide rear view video on the heads-up display

All of it combined together, gives us such a system which is going to help bikers ride much more safely than before, and decreasing accident rates related to bikers.

## Rationale

Cyclists and motorcyclists are often left at the mercy of drivers with bigger vehicles, and in a packed city like Toronto, it is hard for 2 wheelers not to be worried about the fact that the next second they might get hit by a careless driver.

The dedicated lanes for cyclists aren't always present on every street, forcing the riders to take the main lanes.

Stats suggest that the fatality rate in Toronto related to riders is 1.6 per 100,000 people. Which sadly, is worse Chicago, Seattle and San Francisco.

It is a fact that in most of the cases, riders are hit from behind, usually because they take a turn without checking their blind sides, or they're just not aware of a speeding oncoming vehicle from behind.

With this project, Bike Gear Assistant, our aim is to reduce this aforementioned fatality rate.

That is the main driving force behind our original idea, and our motivation.

**'Make roads safer'** for riders, is our main objective, which we aim to achieve with this project.

## **Hardware Requirements**

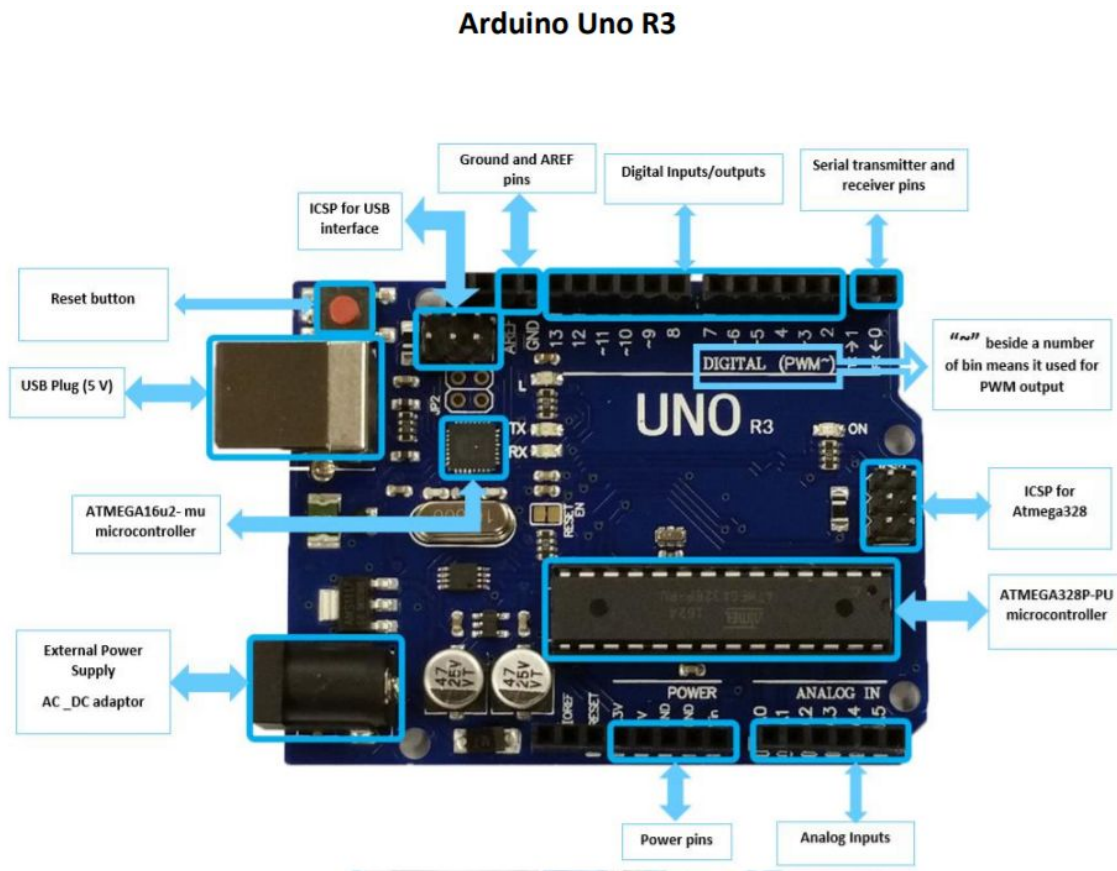
- Raspberry Pi 3 B+
  - To drive the heads-up display system which includes a camera and LCD driving.
- Arduino Uno
  - To drive the ultrasonic sensors array and haptic feedback system.
  - For driving the GSM-Accelerometer system.
- Pi Camera
  - To capture the rear-view in 5MP detail and 16-bit color.
- TFT-LCD 3.5Inch display
  - To display the rear-view video captured by Pi Camera.
- SN-SR04T 5V UltraSonic Sensors
  - For distance detection.
- 2mm Vibrating Disc Motors
  - For vibrational feedback system.
- Convex lens
  - To magnify and produce a virtual image.
- Acrylic glass
  - To project the virtual image onto 3-D space.
- Extension cables
  - For connection flexibility.
- Flex cables
  - For compact connections.

## **OS and Software Requirements**

- Raspbian OS
- Arduino IDE
- QT Creator IDE

## Datasheets and Schematics

### Arduino Uno R3



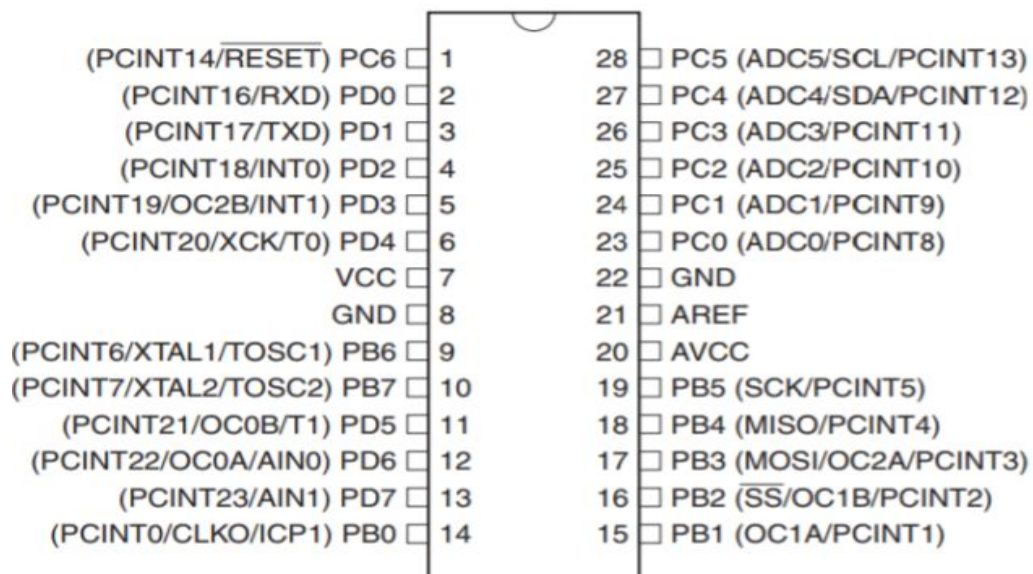
### Details

- Advanced RISC Architecture
  - 32x8 General Purpose Working Registers
  - Upto 20 MIPS throughput at 20MHz
- Non-volatile Memory Segments
  - 4/8/16/32K Bytes of In-System Self-Programmable Flash program memory
  - 256/512/1K Bytes EEPROM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - True Read-While-Write Operation



- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I2 C compatible)
  
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
  
- Power Consumption at 1MHz, 1.8V at 25C
  - Active Mode: 0.2 mA
  - Power-down Mode: 0.1  $\mu$ A
  - Power-save Mode: 0.75  $\mu$ A

#### Pin Diagram



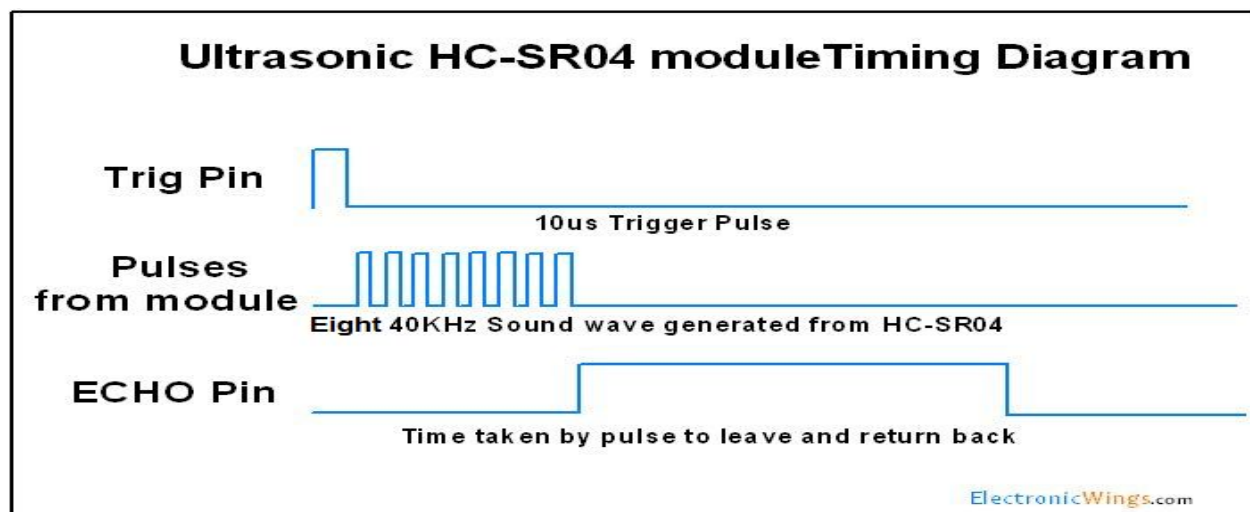
## HC - SR04 Ultrasonic Sensor



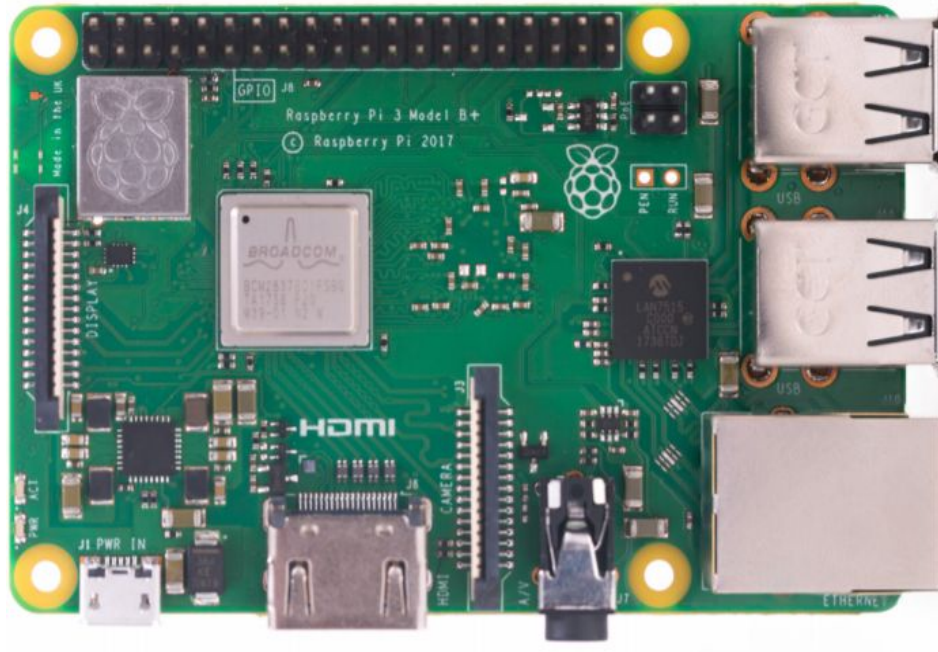
### Details

- 20mm to 700mm non-contact distance and time measurement
- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

### Timing Diagram



## Raspberry Pi 3 B+



### Details

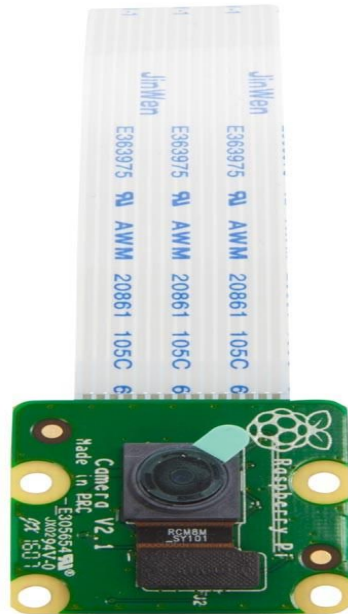
- Processor
  - 64-bit 1.4GHz Broadcom Cortex-A53 processor
- Memory
  - 1GB LPDDR2 SDRAM
- Connectivity
  - 2.4GHz and 5GHz wireless connectivity
  - 4x USB 2.0 ports
  - 1x Full Size HDMI port
  - MIPI CSI Display port
  - MIPI DSI Display port
  - MicroSD card format for loading OS and data storage
- Power rating
  - 5V/2.5A rating
  - 5V through GPIO ports

### *Kuman 3.5 Inches Display*



- Features
  - Supports Raspbian natively
  - Supports video format for 30Hz video playback at 1080p
  - Touch pen for touch interaction
- Specifications
  - TFT LCD
  - 24-bit Color
  - 8:5 Aspect Ratio
  - SPI Interface Protocol

## Pi Camera

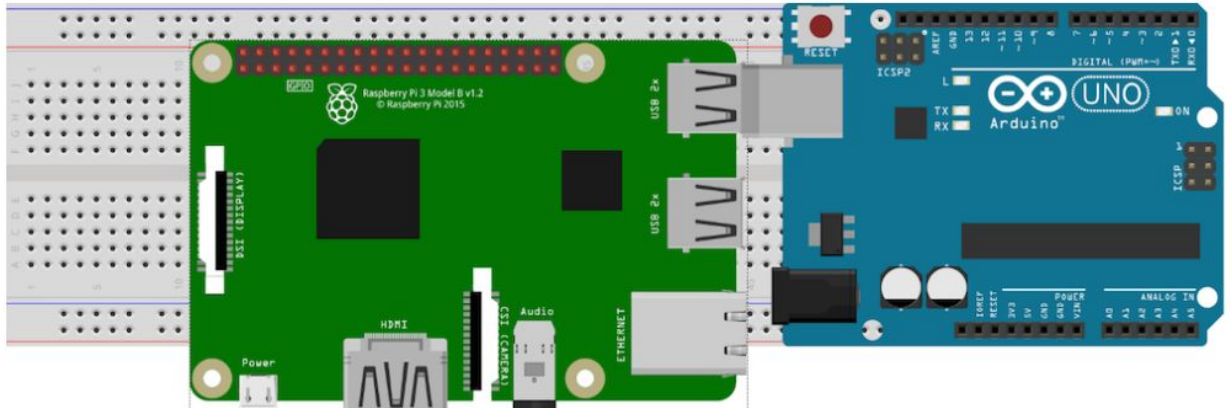


- Features
  - 5 Megapixels
  - 1080p at 30FPS, 720p at 60FPS
- Specifications
  - V4L2 Drivers
  - Omnivision Sensor
  - 2592 x 1994 pixels sensors resolution
  - 3.76mm x 2.74mm sensor image area
  - 1/4 Inch Optical size
  - S/N Ratio 36dB
  - Dynamic Range 67dB at 8x gain
  - 1 meter to infinity fixed focus
  - 3.60mm focal length

The time taken for the sensor to transmit and receive the signal will be used to calculate the distance of the object using a very simple kinematics algorithms.

## 2. Vibrational Feedback Implementation

Arduino will be used to gather the distance and time data captured by the ultrasonic sensor array, and will be used to drive the haptic motors and LEDs



On the other hand, vibrational motors will be interfaced with Uno, and they will work on the basis of the signal processed and received from Uno, which in-turn will depend upon the ultrasonic sensor arrays' arrangement and efficiency.

### Closer the distance, Greater the feedback

This is the method the team will be following for feedback.

The result calculated by Uno will decide if the vibration on the module will be strong or weak.

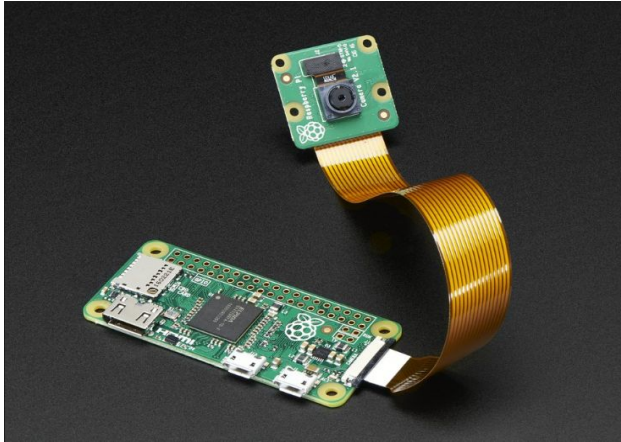
If the object gets closer to the rider, the vibration will get intense, and as the object fades away, so will the vibration.

The vibrational motors, namely 2mm Vibrating Disc Motors, will be interfaced with Arduino Uno ports, and then driven directly using analog pins.

## 3. Video Feedback using Pi Camera and Pi Zero

Working in conjunction with the vibration feedback, will be a camera module, providing real-time feedback to the rider in the heads-up display attached to the helmet.





The reason for choosing Pi Camera and Pi zero over any other combo is that there are in-built libraries in Pi Zero supporting Pi Camera for easy interfacing.

#### 4. Implementation of virtual Heads-up Display

One of the trickiest parts of the project, will be to incorporate the heads-up display, which to be designed in such a way that it doesn't obstruct the rider's view or periphery.

We have decided to use mirror and lens optics to provide a solution for this problem.

Instead of having a physical display in front of the rider, we aim to install a lens which would create a magnified virtual image of the screen in front of the rider, hence causing no obstruction of vision whatsoever.

The following diagram shows an example of what will be incorporated:



This above image shows heads-up display in an aviation helmet's visor. Similar method will be used in our project to incorporate video feedback.



## Working

Working of the project can be divided into 4 separate sections, just like our strategy. Each individual section will describe the functioning of each component related to that section, and at the end will include the co-existent working of the whole system.

### 1. Ultrasonic sensors with Arduino

- **Interfacing-**

Interfacing of these sensors with Arduino is a fairly straightforward task. These sensors have a 4-pin layout, namely TRIG, ECHO, GND and VCC. We created a 4x1 array of ultrasonic sensors to increase the coverage area of sensors, as well as their accuracy.

TRIG and ECHO pins go to the digital pins of Uno, whereas VCC and GND go to static outputs provided by Uno.

- **Initialization-**

Initialization of these requires a 3-step process, which involves bringing the TRIG pin to LOW, pulling TRIG pin to HIGH for 10 microseconds, and reading the value using ECHO pin. Below are the steps with their respective codes.

#### Step-1

*Switching TRIG pin to LOW:*

This step is typically an error prevention step. Since the initial state of the input or output pins should never be assumed, it's always advisable to initialise the pins with default values according to your need.

Following command is used in loop() function to bring the pin to default state.

```
digitalWrite(TRIG_Pin_Name, LOW)
```

Switching the TRIG pin to LOW ensures that the initial state of the pin will be off/low.

#### Step-2

*Switching TRIG pin to HIGH for 10 Microseconds to start sonic transmission:*

The IC on this sensor is designed to react to very specific inputs. In this case, it only starts to transmit ultrasonic frequency bursts when it receives a HIGH signal for specifically 10 *microseconds*. Any value lower or higher than 10 microseconds will either cause a serious error in the output or straight up won't trigger the sensor.

Once the sensor has received a HIGH for 10 microseconds, the TRIG pin is again brought down to LOW to complete the process of *transmission*.

To call the aforementioned step in the code, we use the following commands:

```
digitalWrite(TRIG_Pin_Name, HIGH)
delayMicroseconds(10)
digitalWrite(TRIG_Pin_Name, LOW)
```

When this command is complete, the ultrasonic sensor fires a short burst of ultrasonic frequencies a few microseconds apart. When these pulses hit an object, they reflect back to the sensor where the ECHO pin reads the reflected signal. Reading using the ECHO pin is our next step.

### Step-3

*Switching ECHO pin to HIGH to receive the transmitted ultrasonic signal:*

Once the sonic pulses are reflected back to the sensor, ECHO pin is polled for a HIGH incoming pulse using the *pulseIn* function.

These pulses collected on ECHO pin are essentially the data corresponding to the time taken in microseconds for the sonic pulse to leave the sensor's Transmitter and then reflect back to Receiver.

Following code is used to apply this process:

```
duration= pulseIn(ECHO_Pin_Name, HIGH)
```

- **Utilization of Collected Data-**

Variable 'duration' has been used to store the dynamic values received via ECHO pin.

Dimension of the received data corresponds to *time* dimension, which we need to convert to *length* dimension to gain the necessary values.

We use the distance, speed and time relation to calculate the value of data in microseconds to centimeters.

We use the algorithm  $\text{distance} = \text{duration}/2 * \text{SpeedOfSound}$

The term 'duration' here corresponds to the variable *duration* in our code, which is divided by because the time collected by the sensor is the time difference between the pulse leaving the sensor to the instant when the pulse is received by the sensor. This way the pulse has travelled twice the distance by the time it is received. Hence to compensate for this extra travel time, the total duration is divided by 2.

Then this parameter is multiplied with the speed of sound ie. 343M/s which is also .0343cm/Microsecond. Hence we get the distance of the object in cm which resulted in the reflected sonic pulses.

## 2. Vibration Feedback System

- Interfacing

Interfacing of haptic motors is fairly straightforward to establish. We have incorporated 2 haptic motors, each corresponding to left ultrasonic and right ultrasonic sensor array.

VCC of LEFT haptic motor is connected to PIN 5 of Arduino whereas PIN 6 is used for VCC of RIGHT haptic motor. GND pins of both are connected to GND of Arduino.

```
const int vibration_LEFT    = 5
const int vibration_RIGHT = 6
```

- Operation

Data received by ultrasonic sensors and calculated by Arduino is fed to the timing algorithm of the haptic motors. They are programmed in such a way that if the sensor gathers the data which signifies an object further away, the interval in the haptic output is larger, whereas it becomes shorter as the object reaches closer to the subject.

This is successfully implemented by manipulating the use of *DELAY* command and then using it as a function of distance.

Certain scalars have been used to modify and optimise the *DELAY* command timings to be suitable for our operation.

Example from the code:

```
if ( distance_LEFT>30 && distance_LEFT < 100)
{
    digitalWrite(vibration_LEFT, 255*5);
}
```

```

        delay(200);
        digitalWrite(vibration_LEFT, 0);
        delay(distance_LEFT/2);
    }
else
    {
        digitalWrite(vibration_LEFT, 255*5);
        delay(100);
        digitalWrite(vibration_LEFT, 0);
        delay(distance_LEFT*5);
    }

```

This excerpt shows that the vibration feedback will be of much higher frequency when the object is between 30mm to 100mm from the subject. This frequency has been determined by the parameter *vibration\_LEFT* and *DELAY(distance\_LEFT/2)*.

Similar algorithm has been used for RIGHT sensor feedback.

### 3. LCD Display and Pi camera with Raspberry Pi

- Interfacing LCD

Stated below are the pins used to interface the LCD with Raspberry Pi. Hardware side of the interfacing is fairly straightforward, as shown by the table below.

Pin No:	Symbol	Description
1, 17	3.3V	3.3V power input
2,4	5V	5V power input
3,5,7,8,10,12,13,15,16	NC	No connection – used for support
6,9,14,20,25	GND	Ground
11	TP_IRQ	Active low interrupt pin for touch screen

18	LCD_RS	Register select pin of Display controller
19	LCD_SI	SPI data input for the LCD display
21	TP_SO	SPI data output from the LCD display
22	RST	Reset
23	LCD_SCK	Clock sync pin of SPI communication
24	LCD_CS	Chip select pin of SPI LCD
26	TP_CS	Chip select pin of SPI Touch screen

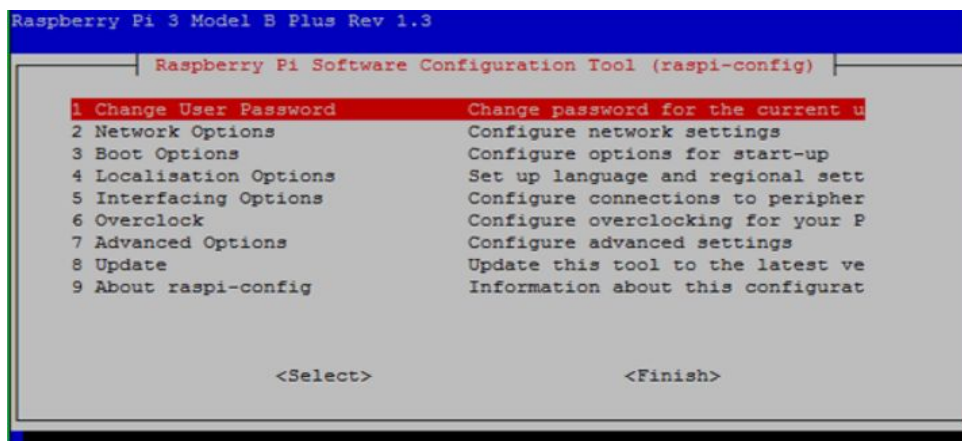
Once the LCD display pins have been connected to the pins of Pi, interfacing on the software side is necessary.

On the processor side, enabling the pins for communication with LCD required gaining access to a library which would enable input/output operation of the processor.

### Configuring and Enabling SPI Interfacing

The display we chose to install uses SPI communication protocol for its operation, so we also needed to enable SPI communication in the main *raspberrypi config*. We can access this config file either using the command ***sudo raspi-config***, or going in the gui menu on the raspberry Pi desktop. We used command line, which shown below:

calling ***sudo raspi-config*** opens this window



We go into *Interfacing Options* and enable *SPI* communication.

### Installing WiringPi

For establishing communication and operation between display and Pi, there is a need to enable the GPIO pins of Pi for input/output operations. This was made possible using the *WiringPi* library.

In newer Raspbian systems, WiringPi is already included as package in the installation image. To check whether the GPIO ports are enabled, we use this command:

```
$ gpio -v
```

If there are any outputs in the command line, that means its already installed and there is no need to re-install the WiringPi package.

In some circumstances we might not want the package installer, but only from the source. In that case, which also applied to us, we used these commands to first delete the existing package and then install a fresh WiringPi package from the source.

```
$ sudo apt-get purge wiringpi
$ hash -r
$ git clone git://git.drogon.net/wiringpi
```

These commands will start the the download of wiringpi from github. To install/build the downloaded package, we used these commands:

```
$ cd ~/wiringPi
$ ./build
```

Once *SPI* is enabled and *WiringPi* is setup for GPIO communication, installing the drivers for the LCD screen are necessary.

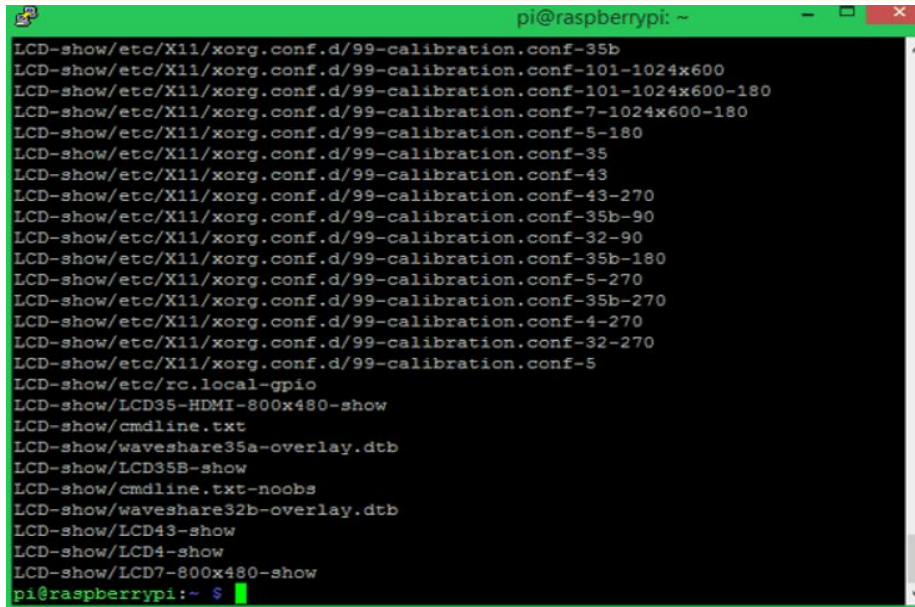
### Installing LCD drivers

To install the drivers, we used the reference site

<https://circuitdigest.com/microcontroller-projects/interfacing-3.5-inch-touchscreen-tft-lcd-with-raspberry-pi>

After downloading the driver package in zip format, we unzipped and installed the package using the following commands, giving the following output:

```
$ tar xvf LCD-show-*.tar.gz
```

A terminal window titled 'pi@raspberrypi: ~' showing the output of the command 'tar xvf LCD-show-\*.tar.gz'. The output lists various files extracted from the archive, including calibration files for different LCD models and resolution combinations, as well as driver files for specific LCDs like LCD35, LCD43, and LCD7. The list ends with 'pi@raspberrypi:~ \$' and a green cursor.

```
pi@raspberrypi: ~
LCD-show/etc/X11/xorg.conf.d/99-calibration.conf-35b
LCD-show/etc/X11/xorg.conf.d/99-calibration.conf-101-1024x600
LCD-show/etc/X11/xorg.conf.d/99-calibration.conf-101-1024x600-180
LCD-show/etc/X11/xorg.conf.d/99-calibration.conf-7-1024x600-180
LCD-show/etc/X11/xorg.conf.d/99-calibration.conf-5-180
LCD-show/etc/X11/xorg.conf.d/99-calibration.conf-35
LCD-show/etc/X11/xorg.conf.d/99-calibration.conf-43
LCD-show/etc/X11/xorg.conf.d/99-calibration.conf-43-270
LCD-show/etc/X11/xorg.conf.d/99-calibration.conf-35b-90
LCD-show/etc/X11/xorg.conf.d/99-calibration.conf-32-90
LCD-show/etc/X11/xorg.conf.d/99-calibration.conf-35b-180
LCD-show/etc/X11/xorg.conf.d/99-calibration.conf-5-270
LCD-show/etc/X11/xorg.conf.d/99-calibration.conf-35b-270
LCD-show/etc/X11/xorg.conf.d/99-calibration.conf-4-270
LCD-show/etc/X11/xorg.conf.d/99-calibration.conf-32-270
LCD-show/etc/X11/xorg.conf.d/99-calibration.conf-5
LCD-show/etc/rc.local-gpio
LCD-show/LCD35-HDMI-800x480-show
LCD-show/cmdline.txt
LCD-show/waveshare35a-overlay.dtb
LCD-show/LCD35B-show
LCD-show/cmdline.txt-noobs
LCD-show/waveshare32b-overlay.dtb
LCD-show/LCD43-show
LCD-show/LCD4-show
LCD-show/LCD7-800x480-show
pi@raspberrypi:~ $
```

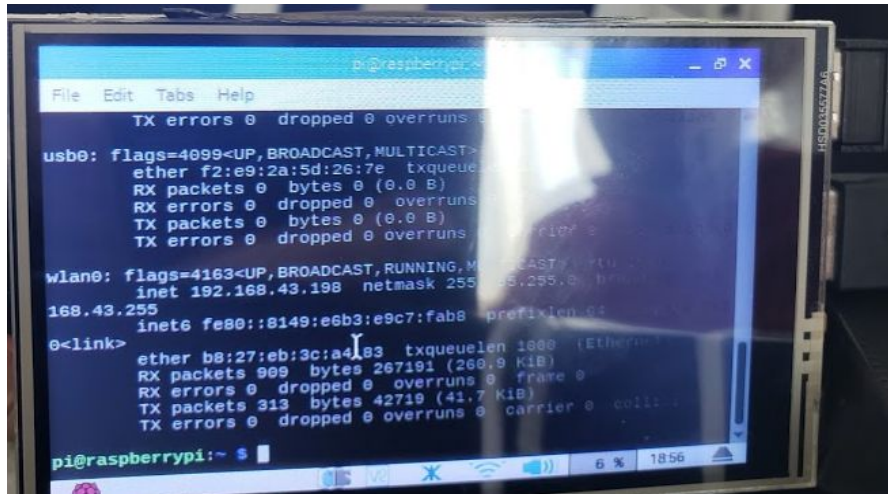
To check if the drivers have been installed, we searched for the driver folder by using following command:

```
cd LCD-show/
```

A terminal window showing the command 'cd LCD-show/' being executed. The prompt changes from 'pi@raspberrypi:~' to 'pi@raspberrypi:~/LCD-show', and a green cursor is visible at the end of the line.

```
pi@raspberrypi:~ $ cd LCD-show/
pi@raspberrypi:~/LCD-show $
```

This means the drivers were installed correctly as the folder exists, and the following is the display you get once everything is set up on the display end.



- Interfacing Pi Camera

Interfacing the camera with Pi doesn't require custom connections, as the camera comes with a ribbon cable directly connectable with Raspberry Pi 2, 3 and 3B as they have dedicated ports for external camera.



BLUE ARROW indicates the port.

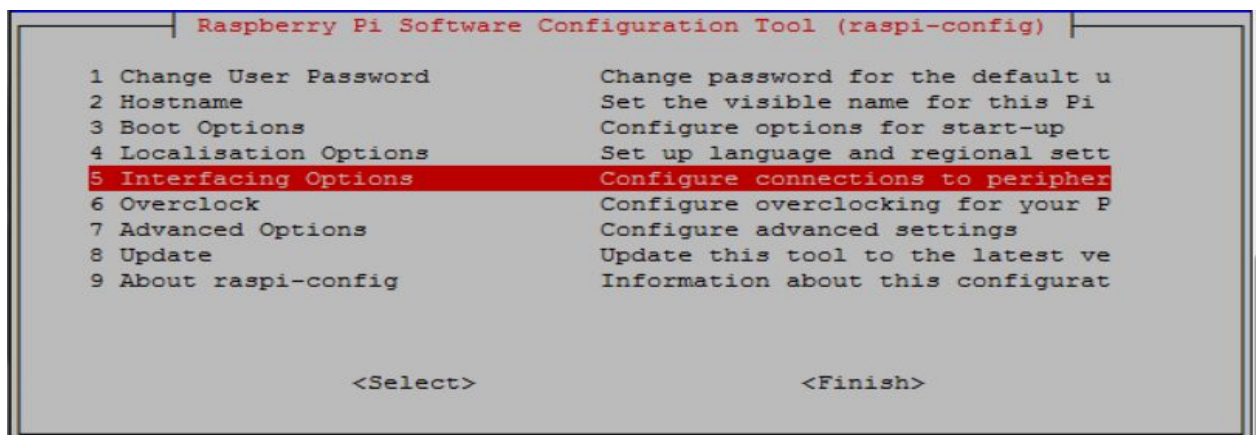


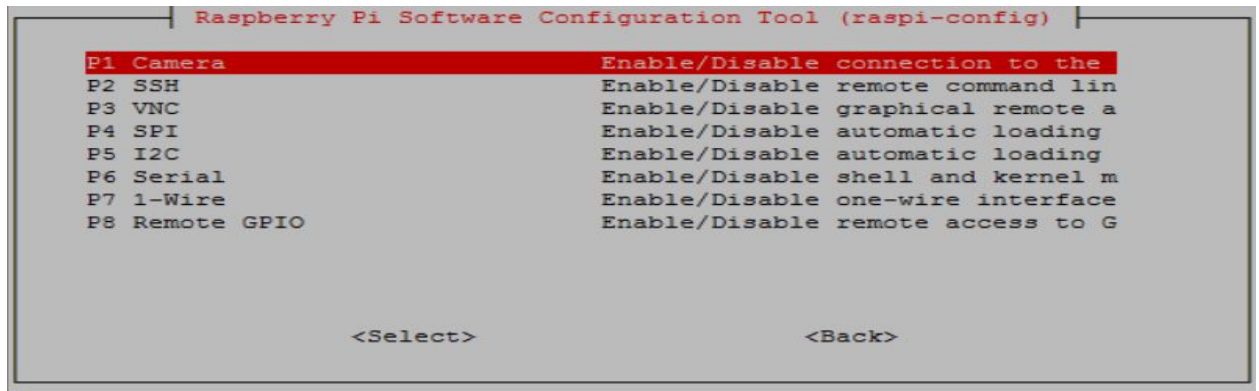


Once the ribbon cable is connected, the hardware side is complete.

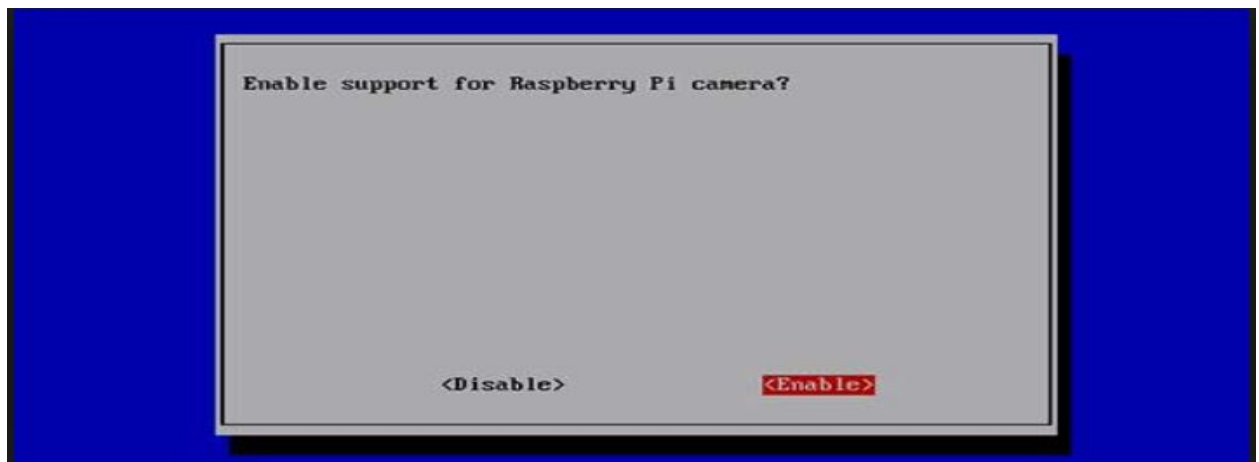
Coming back to the *software side*, we need to enable the access to camera in the *raspberrypi config* file.

So we use **\$sudo raspi-config** command to fetch the config screen, and enable camera by entering the *Interfacing Options* from there. This is how it should look like:





Once in the *Interfacing Options* menu, select **Camera** option to enable it.



- Working of Pi Camera and Display with Raspberry Pi

Purpose of utilizing the display and camera in this project was to incorporate real-time video feedback of rear-view when installed on the helmet. Hence the camera is programmed to work in such a manner that whatever it catches on its sensors, it displays in real-time on the display, without storing it anywhere besides the temporary gpu buffer.

Problems with this approach

Stating the function of displaying the preview of camera on the display seems fairly straightforward. But due to technological limitations of SPI communication protocol, and

also due to size limitations, we faced an unforeseen problem which halted the project's process for a long while.

Camera preview **doesn't work** on SPI displays natively. There is a very specific reason for this issue. But to understand it, we needed to understand how the camera worked with raspberry Pi, and how it displayed previews on regular HDMI displays. Following is the chronology of events which occur( in case of HDMI connection) the moment light falls on the camera:

- ❑ Camera captures the frame
- ❑ Signal is relayed to the Pi via ribbon cable
- ❑ Pi recognizes and processes the signal according to set parameters
- ❑ Pi sends the signal through the gpu, which renders the signal into visual format
- ❑ Gpu buffers the produced frame through HDMI connector to the display

Since we had an SPI display, gpu buffer utilization was out of the question, as there is no connection of gpu with the GPIO ports used for SPI connections. This led us to think of workarounds to have the video rendered on the cpu buffer, so as to display it on the LCD using SPI, instead of getting it rendered on HDMI port.

### Solution

After thorough research about the underlying problem, it was found that there is no alternate way to directly get the camera preview on the SPI display.

Hence the team decided to use an unconventional way to gain access to the preview screen.

We implemented a streaming server on the Pi, have the camera stream the captured frame on the media server, and then get the stream on the Chromium browser on the Pi itself, hence removing the gpu rendering out of the equation for better or worse.

For streaming service, we used the RPi-Cam-Web-Interface. The installation is as follows:

```
$ git clone
https://github.com/silvanmelchior/RPi_Cam_Web_Interface.git
$ cd RPi_Cam_Web_Interface
$ ./install.sh
```

*install.sh* brings upon a menu of multiple choices for server, ip name, device name, connection port etc.

We chose Apache server for our streaming as it is a fairly general server with little to no streaming latency. Once the settings have been initialised and saved, the setup is saved and the script is complete.

Now we run the script using following commands

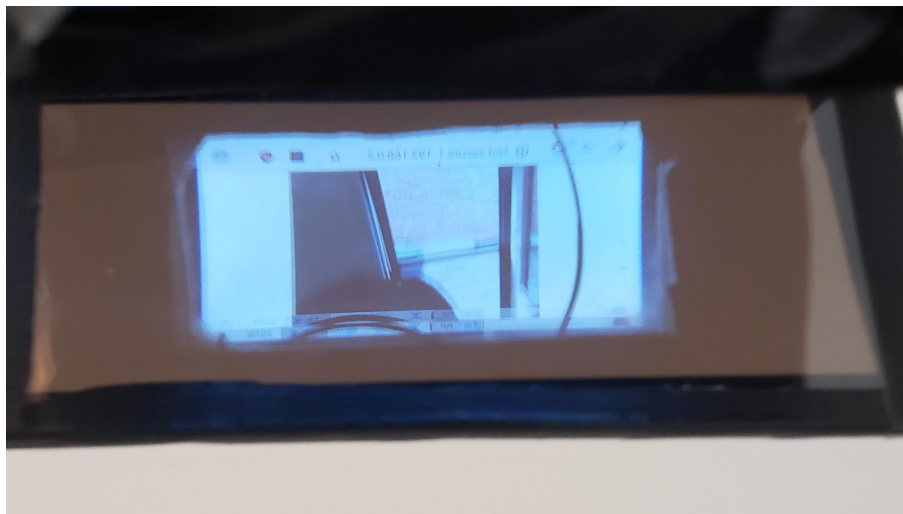
```
$ cd RPi_Cam_Web_Interface  
$ ./start.sh
```

Once these commands are executed, we saw the camera initialized and started, and the streaming server started.

After this customization, this is how the chronology of events go for our camera-display network:

- ☐ Camera captures the frame
- ☐ Signal is sent to Pi using Ribbon cable
- ☐ Pi recognizes the processes the frame
- ☐ Renders it and pushes it through the created media server
- ☐ Chromium is opened on the Pi and ip addresses corresponding to the network is entered
- ☐ The live preview is streamed on the media server

Using this approach, we were able to get a live stream/preview screen of the camera on the SPI display.



#### 4. Heads-up Display Mechanism

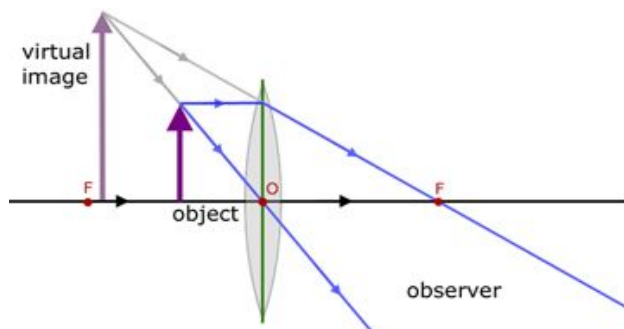
One of the two major features of our project, heads-up display had to be implemented to provide an *unobstructed* view to the rider, while giving a real-time view of the space behind the rider, all projected in the 3D space in front of them.

To make this possible, we had to research on ray optics, and find suitable solutions which could offer a magnified partially reflected image projected onto 3D space in the front.

- Setup and Working

The setup of heads-up mechanism involved the LCD, a magnifying double convex lens, and a partially reflective transparent surface resembling glass.

Magnifying action takes place in a following manner



When an object is kept between the Focus 'F' and Pole 'O', the resultant image is formed behind the lens, which is virtual in nature, magnified in size, and erect is orientation.

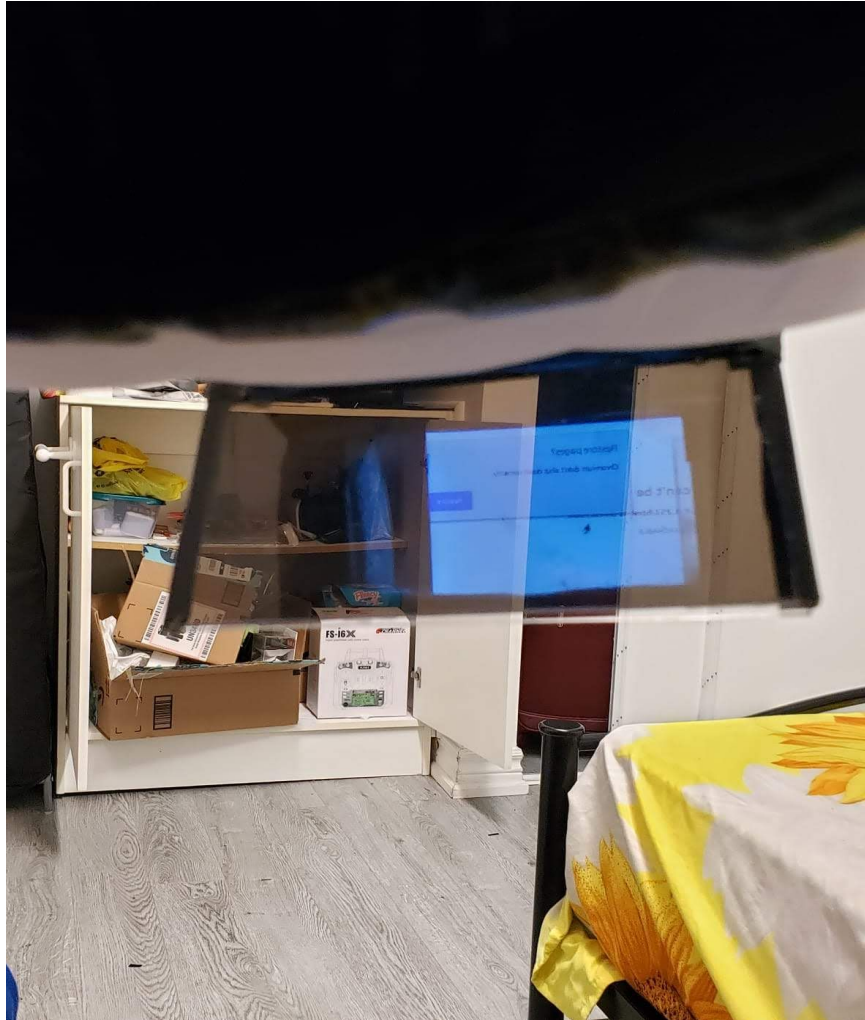
Instead of having an observer in the diagram, if we place a partially reflective plane in front of the lens, we get a resultant which helps us see everything in front of us, while having a virtual image of whatever is behind us (using the camera and lcd), projected onto the 3D space via the transparent partially reflective plane.

Below is a picture of how the final formed image looks:



Here we can see a painting in pink, which is behind the subject, projected in the front while the rest of the view is unobstructed.

Final implementation is presented below, where we slightly tinted the plane surface for better contrast of the projected image.

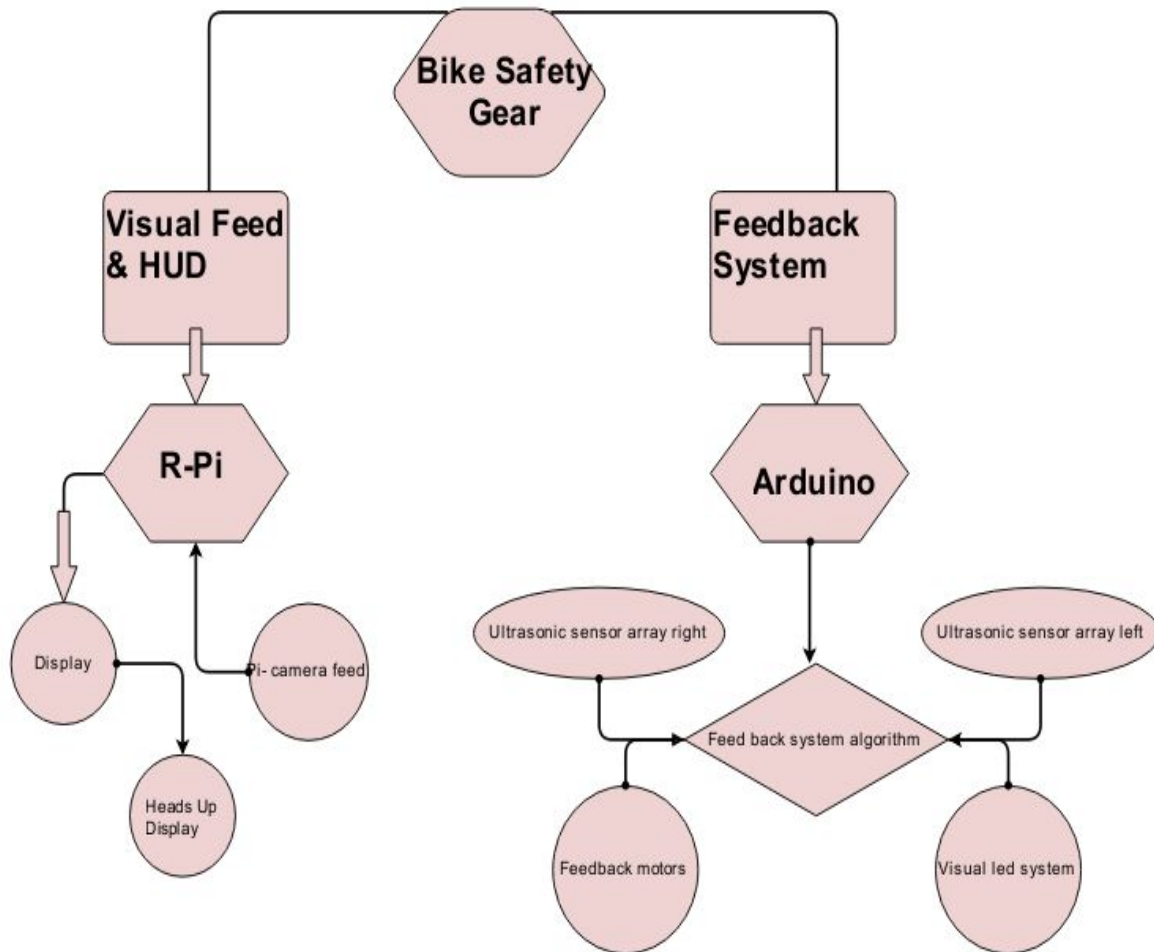


This is an accurate representation of the view of a person wearing the helmet.

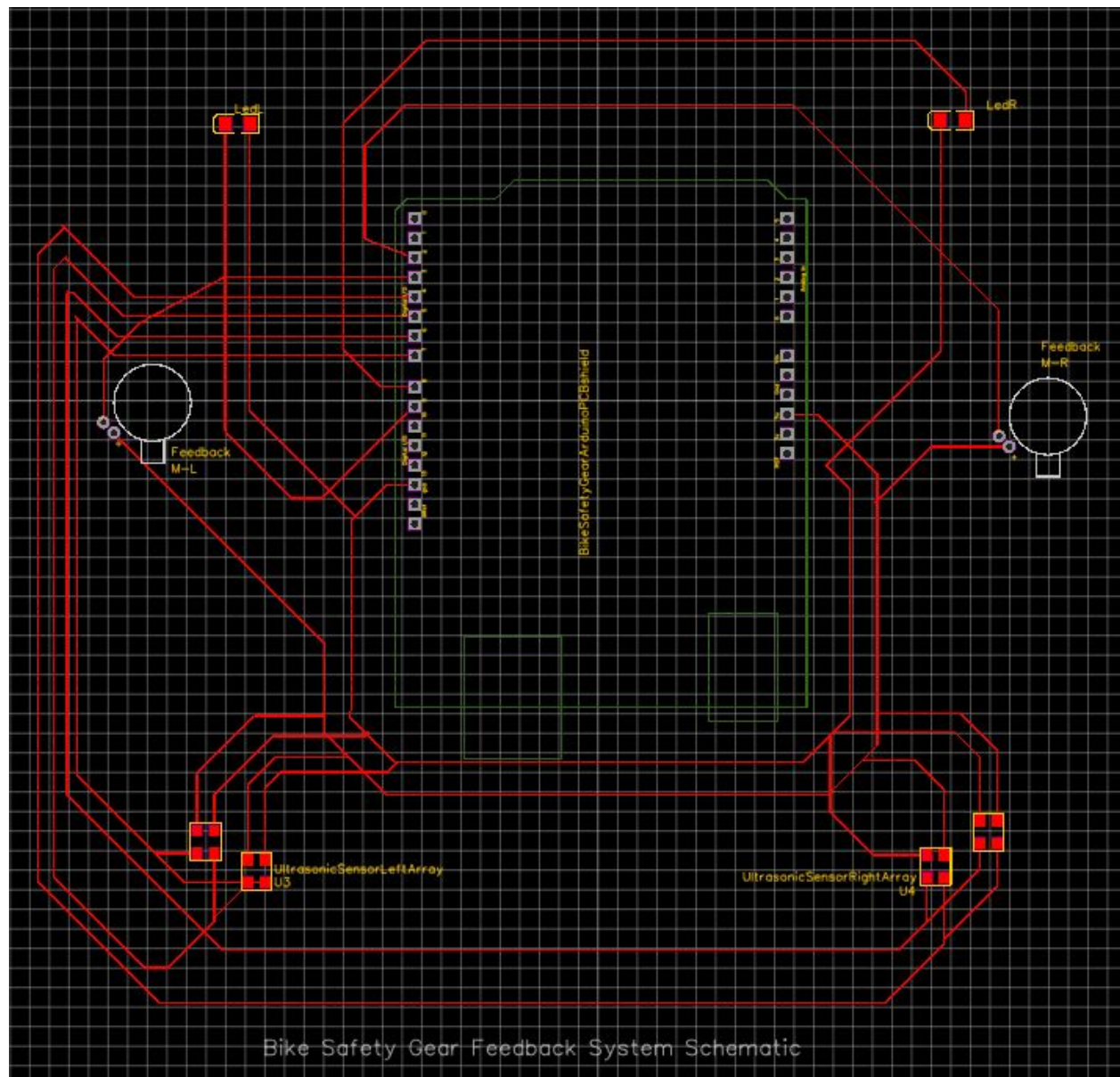
This setup helped us achieve the aim of having a rear view system which doesn't obstruct a person's vision.



## Block Diagram

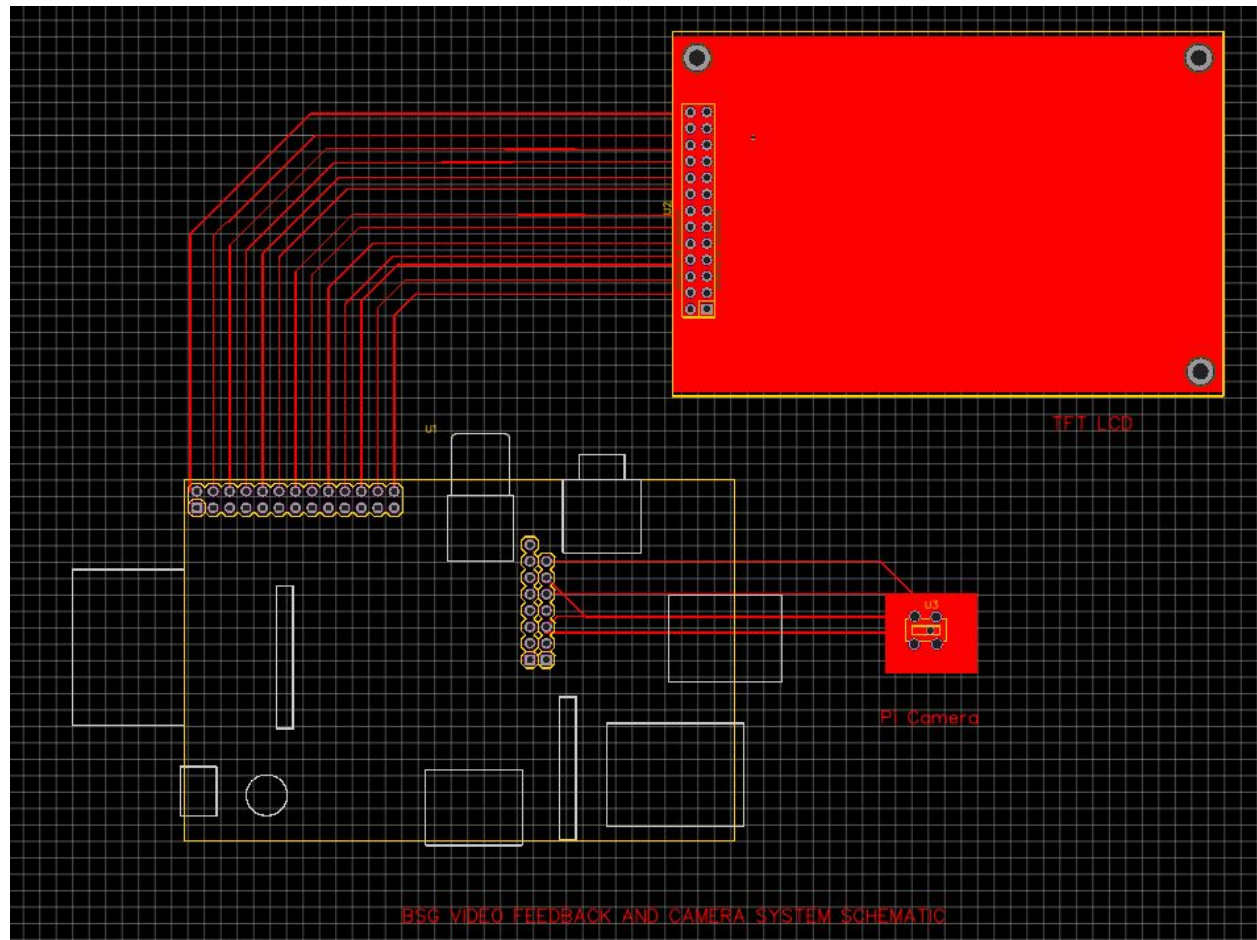


## Schematic Diagrams



BSG feedback system



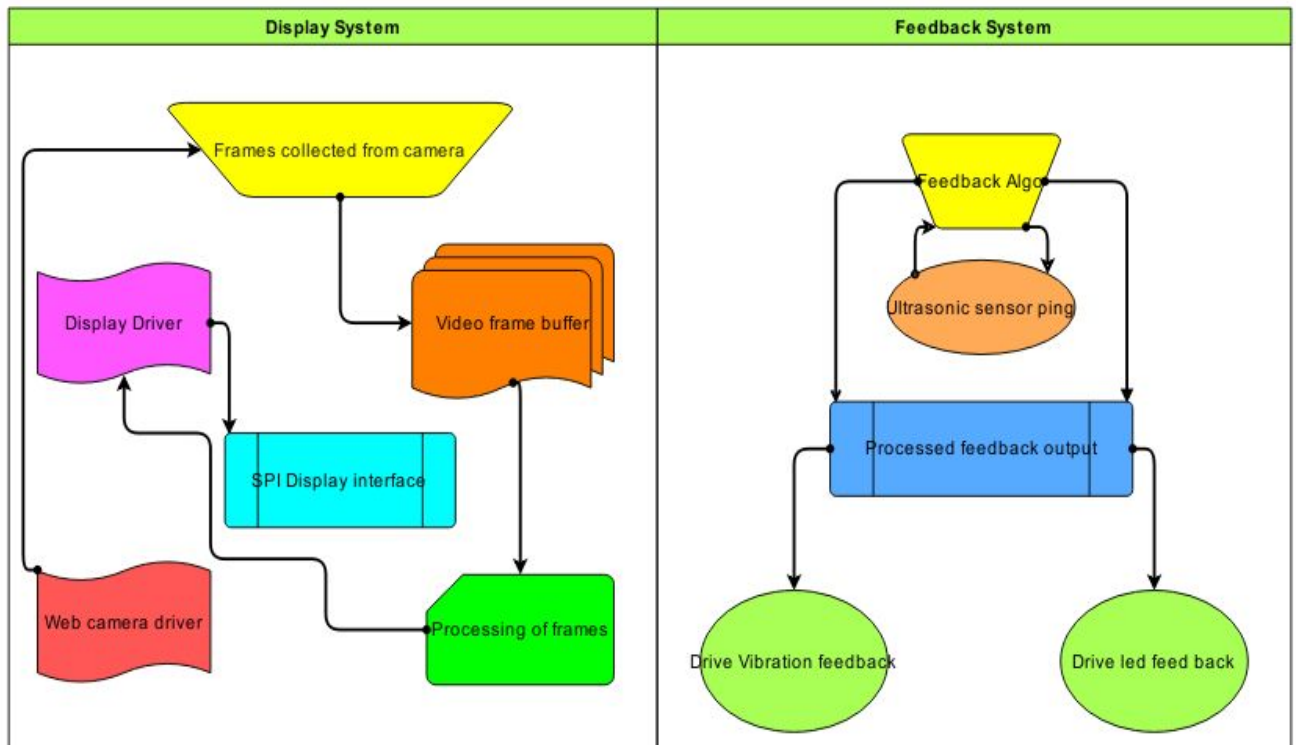


BSG visual feed and HUD system

## PCB Design



## Software Architecture



## Results

By successfully developing this complex feedback system comprising of ultrasonic sensor arrays, leds, haptics and camera-display combo, we were able to produce a product with the following features:

- Object detection outside of the peripheral vision of a rider
- Alerting mechanism comprising of LEDS and Haptic motors
- Rear View video projected right in front of the rider using Heads-up display Mechanism

## Conclusion

*All of the aforementioned functions combined, gives us a device which not only aims to make introduce a new product prototype, but also aims to help the community by making the roads safer via providing riders a '6th Sense' of their surroundings.*



## **References**

### **Arduino Uno and Raspberry Pi interfacing :**

<https://maker.pro/raspberry-pi/tutorial/how-to-connect-and-interface-raspberry-pi-with-arduino>

### **Pi Camera interfacing:**

<https://www.adafruit.com/product/3170>

<https://elinux.org/RPi-Cam-Web-Interface>

### **Pi Camera troubleshooting:**

[https://www.researchgate.net/post/How\\_can\\_one\\_configure\\_Raspberry\\_Pi\\_with\\_Camera\\_and\\_output\\_on\\_computer](https://www.researchgate.net/post/How_can_one_configure_Raspberry_Pi_with_Camera_and_output_on_computer)

### **TFT display drivers:**

[https://www.waveshare.com/wiki/3.5inch\\_RPi\\_LCD\\_\(A\)](https://www.waveshare.com/wiki/3.5inch_RPi_LCD_(A))

### **Camera interface setup:**

<https://mycomputerhelp.net/2015/11/08/minimal-kiosk-browser-kweb-web-browser-fast-reliable-web-browser-for-raspberry-pi-pc/>

### **R-Pi autostart script setup:**

<https://learn.sparkfun.com/tutorials/how-to-run-a-raspberry-pi-program-on-startup/all>

## Appendix

### Ultrasonic sensor and LED Code

*Declaration of ultrasonic sensors, LEDs and haptics:*

```
const int led_RIGHT = 9;
const int led_LEFT = 10;

const int trigPin_LEFT = 3;    //green/blue coupled
const int echoPin_LEFT = 4;    // yellow/yellow coupled
                                // purple vcc, gnd blue // red vcc , brown gnd

const int trigPin_RIGHT = 7;   // orange, R1 = black
const int echoPin_RIGHT = 8;   // red , R1 = Yellow
                                // brown gnd, white vcc // R1= red vcc, Black gnd

const int vibration_LEFT = 5;
      int intensity_LEFT;

const int vibration_RIGHT = 6;
      int intensity_RIGHT;

Long duration_LEFT;
Long duration_RIGHT;
int distance_LEFT;
int distance_RIGHT;

void SonicLEFT();
void SonicRIGHT();
```

*Function definitions for Ultrasonic sensors*

```
void SonicLEFT(void)

{
  digitalWrite(trigPin_LEFT, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin_LEFT, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin_LEFT, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration_LEFT = pulseIn(echoPin_LEFT, HIGH);

  return duration_LEFT;
}
```

```

void SonicRIGHT(void)
{

digitalWrite(trigPin_RIGHT, LOW);
delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin_RIGHT, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin_RIGHT, LOW);
// Reads the echoPin, returns the sound wave travel time in microseconds
duration_RIGHT = pulseIn(echoPin_RIGHT, HIGH);

return duration_RIGHT;
}

```

## *Initialisation and Setup*

```

void setup()
{
pinMode(Led_RIGHT, OUTPUT);
pinMode(Led_LEFT, OUTPUT);

pinMode(vibration_LEFT, OUTPUT);
pinMode(vibration_RIGHT, OUTPUT);

pinMode(trigPin_LEFT, OUTPUT); // Sets the trigPin as an Output
pinMode(trigPin_RIGHT, OUTPUT);

pinMode(echoPin_LEFT, INPUT); // Sets the echoPin as an Input
pinMode(echoPin_RIGHT, INPUT);
Serial.begin(9600); // Starts the serial communication
}

```

## *Function calling and Main Logic*

```

void Loop()
{

SonicLEFT();
SonicRIGHT();

// Calculating the distance
distance_LEFT= duration_LEFT*0.034/2;
distance_RIGHT= duration_RIGHT*0.034/2;

// Prints the distance on the Serial Monitor
Serial.print("Distance LEFT: ");

```

```

Serial.println(distance_LEFT);

Serial.print("Distance RIGHT: ");
Serial.println(distance_RIGHT);

if (distance_LEFT > 150 && distance_LEFT < 500 )
{

    if ( distance_LEFT>150 && distance_LEFT < 200)
    {
        digitalWrite(vibration_LEFT, 255*5);
        digitalWrite(Led_LEFT, 255);
        delay(200);

        digitalWrite(vibration_LEFT, 0);
        digitalWrite(Led_LEFT, 0);
        delay(distance_LEFT/2);
    }

    else
    {
        digitalWrite(vibration_LEFT, 255*5);
        digitalWrite(Led_LEFT, 255);
        delay(100);

        digitalWrite(vibration_LEFT, 0);
        digitalWrite(Led_LEFT, 0);
        delay(distance_LEFT*5);
    }
}

else if (distance_RIGHT > 150 && distance_RIGHT < 500 )
{

    if ( distance_RIGHT>150 && distance_RIGHT < 200)
    {
        digitalWrite(vibration_RIGHT, 255*5);
        digitalWrite(Led_RIGHT, 255);
        delay(200);

        digitalWrite(vibration_RIGHT, 0);
        digitalWrite(Led_RIGHT, 0);
        delay(distance_RIGHT/2);
    }

    else
    {
        digitalWrite(vibration_RIGHT, 255*5);
        digitalWrite(Led_RIGHT, 255);
        delay(100);
    }
}

```



```

        digitalWrite(vibration_RIGHT, 0);
        digitalWrite(led_RIGHT, 0);
        delay(distance_RIGHT*5);

    }

}

}

```

## LCD Driver and Setup (From github)

### *Boot Kernel Config File for 3.5Inch LCD*

```

#hdmi_safe=1

# uncomment this if your display has a black border of unused pixels visible
# and your display can output without overscan
#disable_overscan=1

# uncomment the following to adjust overscan. Use positive numbers if console
# goes off screen, and negative if there is too much border
#overscan_left=16
#overscan_right=16
#overscan_top=16
#overscan_bottom=16

# uncomment to force a console size. By default it will be display's size minus
# overscan.
#framebuffer_width=1280
#framebuffer_height=720

# uncomment if hdmi display is not detected and composite is being output
hdmi_force_hotplug=1

# uncomment to force a specific HDMI mode (this will force VGA)
#hdmi_group=1
#hdmi_mode=1

# uncomment to force a HDMI mode rather than DVI. This can make audio work in
# DMT (computer monitor) modes
#hdmi_drive=2

# uncomment to increase signal to HDMI, if you have interference, blanking, or
# no display
#config_hdmi_boost=4

# uncomment for composite PAL
#sdtv_mode=2

#uncomment to overClock the arm. 700 MHz is the default.
#arm_freq=800

# Uncomment some or all of these to enable the optional hardware interfaces

```

```

dtparam=i2c_arm=on
#dtparam=i2s=on
dtparam=spi=on
enable_uart=1
# Uncomment this to enable the lirc-rpi module
#dtoverlay=lirc-rpi

# Additional overlays and parameters are documented /boot/overlays/README

# Enable audio (loads snd_bcm2835)
dtparam=audio=on
dtoverlay=tft35a
#dtoverlay=ads7846,cs=1,penirq=17,penirq_pull=2,speed=1000000,keep_vref_on=1,swapxy=1,pmax=255,xohms=60,
xmin=200,xmax=3900,ymin=200,ymax=3900

```

## LCD Initialisation and display Code

```

#!/bin/bash
sudo rm -rf /etc/X11/xorg.conf.d/40-libinput.conf
sudo mkdir /etc/X11/xorg.conf.d
sudo cp ./usr/tft35a-overlay.dtb /boot/overlays/
sudo cp ./usr/tft35a-overlay.dtb /boot/overlays/tft35a.dtbo
sudo cp -rf ./usr/99-calibration.conf-35 /etc/X11/xorg.conf.d/99-calibration.conf
sudo cp -rf ./usr/99-fbturbo.conf /usr/share/X11/xorg.conf.d/
if [ -b /dev/mmcblk0p7 ]; then
sudo cp ./usr/cmdline.txt-noobs /boot/cmdline.txt
else
sudo cp ./usr/cmdline.txt /boot/
fi
sudo cp ./usr/inittab /etc/
sudo cp ./boot/config-35.txt /boot/config.txt
nodeplatform=`uname -n`
kernel=`uname -r`
version=`uname -v`
if test "$nodeplatform" = "raspberrypi";then
echo "this is raspberrypi kernel"
version=${version%% *}
version=${version##*#}
echo $version
if test $version -lt 970;then
echo "reboot"
else
echo "need to update touch configuration"
sudo apt-get install xserver-xorg-input-evdev
sudo cp -rf /usr/share/X11/xorg.conf.d/10-evdev.conf /usr/share/X11/xorg.conf.d/45-evdev.conf
echo "reboot"
fi
else
echo "this is not raspberrypi kernel, no need to update touch configure, reboot"
fi
sudo reboot

```

## Rpi\_Cam\_Web Camera Stream Library Setup

### *./install.sh script*

```
exec 5> install.txt
BASH_XTRACEFD="5"
set -x

cd $(dirname $(readlink -f $0))

if [ $(dpkg-query -W -f='${Status}' "dialog" 2>/dev/null | grep -c "ok installed") -eq 0 ]; then
    sudo apt-get install -y dialog
fi

# Terminal colors
color_red="tput setaf 1"
color_green="tput setaf 2"
color_reset="tput sgr0"

# Version stuff moved out functions as we need it more when one time.
versionfile="./www/config.php"
version=$(cat $versionfile | grep "'APP_VERSION'" | cut -d '"' -f4)
backtitle="Copyright (c) 2015, Bob Tidey. RPi Cam $version"
jpglink="no"
phpversion=7

# Config options located in ./config.txt. In first run script makes that file for you.
if [ ! -e ./config.txt ]; then
    sudo echo "#This is config file for main installer. Put any extra options in here." > ./config.txt
    sudo echo "rpicamdir=\"html\"" >> ./config.txt
    sudo echo "webserver=\"apache\"" >> ./config.txt
    sudo echo "webport=\"80\"" >> ./config.txt
    sudo echo "user=\"\"\" >> ./config.txt
    sudo echo "webpasswd=\"\"\" >> ./config.txt
    sudo echo "autostart=\"yes\"" >> ./config.txt
    sudo echo "jpglink=\"no\"" >> ./config.txt
    sudo echo "phpversion=\"7\"" >> ./config.txt
    sudo echo "" >> ./config.txt
    sudo chmod 664 ./config.txt
fi

source ./config.txt
rpicamdiroid=$rpicamdir
if [ ! "${rpicamdiroid:0:1}" == "/" ]; then
    rpicamdiroid=/$rpicamdiroid
fi

#Allow for a quiet install
rm exitfile.txt >/dev/null 2>&1
if [ $# -eq 0 ] || [ "$1" != "q" ]; then
    exec 3>&1
    dialog \
```

```

--separate-widget $'\n'
--title "Configuration Options"
--backtitle "$backtitle"
--form ""
0 0 0
"Cam subfolder:"      1 1  "$rpcamdir"  1 32 15 0 \
"Autostart:(yes/no)"  2 1  "$autostart"  2 32 15 0 \
"Server:(apache/nginx/Lighttpd)" 3 1  "$webserver"  3 32 15 0 \
"Webport:"            4 1  "$webport"    4 32 15 0 \
"User:(blank=nologin)" 5 1  "$user"        5 32 15 0 \
"Password:"           6 1  "$webpasswd"  6 32 15 0 \
"jpglink:(yes/no)"    7 1  "$jpglink"    7 32 15 0 \
"phpversion:(5/7)"    8 1  "$phpversion"  8 32 15 0 \
2>&1 1>&3 | {
    read -r rpcamdir
    read -r autostart
    read -r webserver
    read -r webport
    read -r user
    read -r webpasswd
    read -r jpglink
    read -r phpversion
    if [ -n "$webport" ]; then
        sudo echo "#This is edited config file for main installer. Put any extra options in here." >
./config.txt
        sudo echo "rpcamdir=\"$rpcamdir\" >> ./config.txt
        sudo echo "webserver=\"$webserver\" >> ./config.txt
        sudo echo "webport=\"$webport\" >> ./config.txt
        sudo echo "user=\"$user\" >> ./config.txt
        sudo echo "webpasswd=\"$webpasswd\" >> ./config.txt
        sudo echo "autostart=\"$autostart\" >> ./config.txt
        sudo echo "jpglink=\"$jpglink\" >> ./config.txt
        sudo echo "phpversion=\"$phpversion\" >> ./config.txt
        sudo echo "" >> ./config.txt
    else
        echo "exit" > ./exitfile.txt
    fi
}
exec 3>&-

if [ -e exitfile.txt ]; then
    rm exitfile.txt
    exit
fi

source ./config.txt
fi

if [ ! "${rpcamdir:0:1}" == "" ]; then
    rpcamdir=${rpcamdir}
    rpcamdirEsc=${rpcamdir//\\/\}
else
    rpcamdirEsc=""
fi

fn_stop ()

```

```

{ # This is function stop
    sudo killall raspimjpeg 2>/dev/null
    sudo killall php 2>/dev/null
    sudo killall motion 2>/dev/null
}

fn_reboot ()
{ # This is function reboot system
    dialog --title "Start camera system now" --backtitle "$backtitle" --yesno "Start now?" 5 33
    response=$?
    case $response in
        0) ./start.sh;;
        1) dialog --title 'Start or Reboot message' --colors --infobox "\Zb\Z1"'Manually run ./start.sh or
reboot!' 4 28 ; sleep 2;;
        255) dialog --title 'Start or Reboot message' --colors --infobox "\Zb\Z1"'Manually run ./start.sh
or reboot!' 4 28 ; sleep 2;;
        esac
    }

fn_apache ()
{
aconf="/etc/apache2/sites-available/raspicam.conf"
cp $aconf.1 $aconf
if [ -e "$aconf" ]; then
    sudo rm "$aconf"
fi
if [ -e /etc/apache2/conf-available/other-vhosts-access-log.conf ]; then
    aotherLog="/etc/apache2/conf-available/other-vhosts-access-log.conf"
else
    aotherLog="/etc/apache2/conf.d/other-vhosts-access-log"
fi
tmpfile=$(mktemp)
sudo awk '/NameVirtualHost \*:{c+=1}{if(c==1){sub("NameVirtualHost \*:.*", "NameVirtualHost
*:'$webport'", $0)}};print}' /etc/apache2/ports.conf > "$tmpfile" && sudo mv "$tmpfile"
/etc/apache2/ports.conf
sudo awk '/Listen/{c+=1}{if(c==1){sub("Listen.*", "Listen '$webport'", $0)}};print}'
/etc/apache2/ports.conf > "$tmpfile" && sudo mv "$tmpfile" /etc/apache2/ports.conf
awk '/<VirtualHost \*:{c+=1}{if(c==1){sub("<VirtualHost \*:.*", "<VirtualHost
*:'$webport'>", $0)}};print}' $aconf > "$tmpfile" && sudo mv "$tmpfile" $aconf
sudo sed -i "s/<Directory\ \\/var\/www\/.*<Directory\ \\/var\/www\$rpicamdirEsc>/g" $aconf
if [ "$user" == "" ]; then
    sudo sed -i "s/AllowOverride\ \\/.*AllowOverride None/g" $aconf
else
    if [ ! -e /usr/local/.htpasswd ]; then
        sudo htpasswd -b -B -c /usr/local/.htpasswd $user $webpasswd
    fi
    sudo htpasswd -b -B /usr/local/.htpasswd $user $webpasswd
    sudo sed -i "s/AllowOverride\ \\/.*AllowOverride ALL/g" $aconf
    if [ ! -e /var/www/$rpicamdir/.htaccess ]; then
        sudo bash -c "cat > /var/www/$rpicamdir/.htaccess" << EOF
AuthName "RPi Cam Web Interface Restricted Area"
AuthType Basic
AuthUserFile /usr/local/.htpasswd
Require valid-user
EOF

```

```

        sudo chown -R www-data:www-data /var/www${rplicamdir}/.htaccess
    fi
fi
sudo mv ${aconf} /${aconf}
if [ ! -e /etc/apache2/sites-enabled/raspicam.conf ]; then
    sudo ln -sf /${aconf} /etc/apache2/sites-enabled/raspicam.conf
fi
sudo sed -i 's/^CustomLog/#CustomLog/g' $aotherLog
sudo a2dissite 000-default.conf >/dev/null 2>&1
sudo service apache2 restart
}

fn_nginx ()
{
    aconf="etc/nginx/sites-available/rpicam"
    cp ${aconf}.1 ${aconf}
    if [ -e "${aconf}" ]; then
        sudo rm "${aconf}"
    fi
    #uncomment next line if wishing to always access by http://ip as the root
    #sudo sed -i "s:root /var/www::root /var/www${rplicamdirEsc};g" ${aconf}
    sudo mv /etc/nginx/sites-available/*default* etc/nginx/sites-available/ >/dev/null 2>&1

    if [ "$user" == "" ]; then
        sed -i "s/auth_basic\ .*/auth_basic \"Off\";/g" ${aconf}
        sed -i "s/\ auth_basic_user_file/#auth_basic_user_file/g" ${aconf}
    else
        sudo htpasswd -b -c /usr/local/.htpasswd $user $webpasswd
        sed -i "s/auth_basic\ .*/auth_basic \"Restricted\";/g" ${aconf}
        sed -i "s/#auth_basic_user_file/\ auth_basic_user_file/g" ${aconf}
    fi
    if [[ "$phpversion" == "7" ]]; then
        sed -i "s/\var/run/php5-fpm.sock;/run/php/php7.0-fpm.sock;/g" ${aconf}
    fi
    sudo mv ${aconf} /${aconf}
    sudo chmod 644 /${aconf}
    if [ ! -e /etc/nginx/sites-enabled/rpicam ]; then
        sudo ln -sf /${aconf} /etc/nginx/sites-enabled/rpicam
    fi

    # Update nginx main config file
    sudo sed -i "s/worker_processes 4;/worker_processes 2;/g" /etc/nginx/nginx.conf
    sudo sed -i "s/worker_connections 768;/worker_connections 128;/g" /etc/nginx/nginx.conf
    sudo sed -i "s/gzip on;/gzip off;/g" /etc/nginx/nginx.conf
    if [ "$NGINX_DISABLE_LOGGING" != "" ]; then
        sudo sed -i "s:access_log /var/log/nginx/nginx/access.log::access_log /dev/null::g"
        /etc/nginx/nginx.conf
    fi

    # Configure php-apc
    if [[ "$phpversion" == "7" ]]; then
        phpnv=/etc/php/7.0
    else
        phpnv=/etc/php5
    fi
    sudo sh -c "echo \"cgi.fix_pathinfo = 0;\" >> $phpnv/fpm/php.ini"

```

```

sudo mkdir $phpnv/conf.d >/dev/null 2>&1
sudo cp etc/php5/apc.ini $phpnv/conf.d/20-apc.ini
sudo chmod 644 $phpnv/conf.d/20-apc.ini
sudo service nginx restart
}

```

**fn\_lighttpd ()**

```

{
sudo lighty-enable-mod fastcgi-php
sudo sed -i "s/^server.document-root.*/server.document-root = \"\|var\|www$rpicamdirEsc\|/g"
/etc/Lighttpd/Lighttpd.conf
sudo sed -i "s/^server.port.*/server.port = $webport/g" /etc/Lighttpd/lighttpd.conf
#sudo service lighttpd restart
sudo /etc/init.d/lighttpd force-reload
}

```

**fn\_motion ()**

```

{
sudo sed -i "s/^daemon.*/daemon on/g" /etc/motion/motion.conf
sudo sed -i "s/^logfile.*/logfile \|tmp\|motion.log /g" /etc/motion/motion.conf
sudo sed -i "s/^; netcam_url.*/netcam_url/g" /etc/motion/motion.conf
sudo sed -i "s/^netcam_url.*/netcam_url http:\|\/localhost:$webport$rpicamdirEsc\|cam_pic.php/g"
/etc/motion/motion.conf
if [ "$user" == "" ]; then
sudo sed -i "s/^netcam_userpass.*/; netcam_userpass value/g" /etc/motion/motion.conf
else
sudo sed -i "s/^; netcam_userpass.*/netcam_userpass/g" /etc/motion/motion.conf
sudo sed -i "s/^netcam_userpass.*/netcam_userpass $user:$webpasswd/g" /etc/motion/motion.conf

```

**fi**

```

sudo sed -i "s/^; on_event_start.*/on_event_start/g" /etc/motion/motion.conf
sudo sed -i "s/^on_event_start.*/on_event_start echo -n \|1\| >\|var\|www$rpicamdirEsc\|FIFO1/g"
/etc/motion/motion.conf
sudo sed -i "s/^; on_event_end.*/on_event_end/g" /etc/motion/motion.conf
sudo sed -i "s/^on_event_end.*/on_event_end echo -n \|0\| >\|var\|www$rpicamdirEsc\|FIFO1/g"
/etc/motion/motion.conf
sudo sed -i "s/control_port.*/control_port 6642/g" /etc/motion/motion.conf
sudo sed -i "s/control_html_output.*/control_html_output off/g" /etc/motion/motion.conf
sudo sed -i "s/^output_pictures.*/output_pictures off/g" /etc/motion/motion.conf
sudo sed -i "s/^ffmpeg_output_movies on/ffmpeg_output_movies off/g" /etc/motion/motion.conf

sudo sed -i "s/^ffmpeg_cap_new on/ffmpeg_cap_new off/g" /etc/motion/motion.conf
sudo sed -i "s/^stream_port.*/stream_port 0/g" /etc/motion/motion.conf
sudo sed -i "s/^webcam_port.*/webcam_port 0/g" /etc/motion/motion.conf
sudo sed -i "s/^process_id_file;/ process_id_file/g" /etc/motion/motion.conf
sudo sed -i "s/^videodevice;/ videodevice/g" /etc/motion/motion.conf
sudo sed -i "s/^event_gap 60/event_gap 3/g" /etc/motion/motion.conf
sudo chown motion:www-data /etc/motion/motion.conf
sudo chmod 664 /etc/motion/motion.conf
}

```

**fn\_autostart ()**

```

{
tmpfile=$(mktemp)
sudo sed '/#START/,/#END/d' /etc/rc.local > "$tmpfile" && sudo mv "$tmpfile" /etc/rc.local
# Remove to growing plank lines.

```

```

sudo awk '!NF {if (++n <= 1) print; next}; {n=0;print}}' /etc/rc.Local > "$tmpfile" && sudo mv "$tmpfile"
/etc/rc.Local
if [ "$autostart" == "yes" ]; then
    if ! grep -Fq '#START RASPIMJPEG SECTION' /etc/rc.Local; then
        sudo sed -i '/exit 0/d' /etc/rc.Local
        sudo bash -c "cat >> /etc/rc.Local" << EOF
#START RASPIMJPEG SECTION
mkdir -p /dev/shm/mjpeg
chown www-data:www-data /dev/shm/mjpeg
chmod 777 /dev/shm/mjpeg
sleep 4;su -c 'raspimjpeg > /dev/null 2>&1 &' www-data
if [ -e /etc/debian_version ]; then
    sleep 4;su -c 'php /var/www$rpicamdir/schedule.php > /dev/null 2>&1 &' www-data
else
    sleep 4;su -s '/bin/bash' -c 'php /var/www$rpicamdir/schedule.php > /dev/null 2>&1 &' www-data
fi
#END RASPIMJPEG SECTION

exit 0
EOF
else
    tmpfile=$(mktemp)
    sudo sed '/#START/,/#END/d' /etc/rc.Local > "$tmpfile" && sudo mv "$tmpfile" /etc/rc.Local
    # Remove to growing plank lines.
    sudo awk '!NF {if (++n <= 1) print; next}; {n=0;print}}' /etc/rc.Local > "$tmpfile" && sudo mv
"$tmpfile" /etc/rc.Local
fi

fi
sudo chown root:root /etc/rc.Local
sudo chmod 755 /etc/rc.Local
}

#Main install)
fn_stop

sudo mkdir -p /var/www$rpicamdir/media
#move old material if changing from a different install folder
if [ ! "$rpicamdir" == "$rpicamdirold" ]; then
    if [ -e /var/www$rpicamdirold/index.php ]; then
        sudo mv /var/www$rpicamdirold/* /var/www$rpicamdir
    fi
fi

sudo cp -r www/* /var/www$rpicamdir/
if [ -e /var/www$rpicamdir/index.html ]; then
    sudo rm /var/www$rpicamdir/index.html
fi

if [ "$phpversion" == "7" ]; then
    phpv=php7.0
else
    phpv=php5
fi

if [ "$webserver" == "apache" ]; then

```



```

sudo apt-get install -y apache2 $phpv $phpv-cli libapache2-mod-$phpv gpac motion zip libav-tools
gststreamer1.0-tools
    if [ $? -ne 0 ]; then exit; fi
    fn_apache
elif [ "$webserver" == "nginx" ]; then
    sudo apt-get install -y nginx $phpv-fpm $phpv-cli $phpv-common $phpv-apcu apache2-utils gpac motion
    zip libav-tools gststreamer1.0-tools
    if [ $? -ne 0 ]; then exit; fi
    fn_nginx
elif [ "$webserver" == "lighttpd" ]; then
    sudo apt-get install -y lighttpd $phpv-cli $phpv-common $phpv-cgi $phpv gpac motion zip libav-tools
    gststreamer1.0-tools
    if [ $? -ne 0 ]; then exit; fi
    fn_lighttpd
fi

#Make sure user www-data has bash shell
sudo sed -i "s/^www-data:x.*:/www-data:x:33:33:www-data:\/var\/www:\/bin\/bash/g" /etc/passwd

if [ ! -e /var/www$rpicamdir/FIFO ]; then
    sudo mknod /var/www$rpicamdir/FIFO p
fi
sudo chmod 666 /var/www$rpicamdir/FIFO

if [ ! -e /var/www$rpicamdir/FIFO11 ]; then
    sudo mknod /var/www$rpicamdir/FIFO11 p
fi
sudo chmod 666 /var/www$rpicamdir/FIFO11

if [ ! -e /var/www$rpicamdir/FIFO1 ]; then
    sudo mknod /var/www$rpicamdir/FIFO1 p
fi

sudo chmod 666 /var/www$rpicamdir/FIFO1

if [ ! -d /dev/shm/mjpeg ]; then
    mkdir /dev/shm/mjpeg
fi

if [ "$jpglink" == "yes" ]; then
    if [ ! -e /var/www$rpicamdir/cam.jpg ]; then
        sudo ln -sf /dev/shm/mjpeg/cam.jpg /var/www$rpicamdir/cam.jpg
    fi
fi

if [ -e /var/www$rpicamdir/status_mjpeg.txt ]; then
    sudo rm /var/www$rpicamdir/status_mjpeg.txt
fi
if [ ! -e /dev/shm/mjpeg/status_mjpeg.txt ]; then
    echo -n 'halted' > /dev/shm/mjpeg/status_mjpeg.txt
fi
sudo chown www-data:www-data /dev/shm/mjpeg/status_mjpeg.txt
sudo ln -sf /dev/shm/mjpeg/status_mjpeg.txt /var/www$rpicamdir/status_mjpeg.txt

sudo chown -R www-data:www-data /var/www$rpicamdir
sudo cp etc/sudoers.d/RPI_Cam_Web_Interface /etc/sudoers.d/

```

```

sudo chmod 440 /etc/sudoers.d/RPI_Cam_Web_Interface

sudo cp -r bin/raspimjpeg /opt/vc/bin/
sudo chmod 755 /opt/vc/bin/raspimjpeg
if [ ! -e /usr/bin/raspimjpeg ]; then
    sudo ln -s /opt/vc/bin/raspimjpeg /usr/bin/raspimjpeg
fi

sed -e "s/www/www${rpicamdirEsc}/" etc/raspimjpeg/raspimjpeg.1 > etc/raspimjpeg/raspimjpeg
if [[ `cat /proc/cmdline |awk -v RS=' ' -F= '/boardrev/ { print $2 }'` == "0x11" ]]; then
    sed -i 's/^camera_num 0/camera_num 1/g' etc/raspimjpeg/raspimjpeg
fi
if [ -e /etc/raspimjpeg ]; then
    $color_green; echo "Your custom raspimjpg backed up at /etc/raspimjpeg.bak"; $color_reset
    sudo cp -r /etc/raspimjpeg /etc/raspimjpeg.bak
fi
sudo cp -r etc/raspimjpeg/raspimjpeg /etc/
sudo chmod 644 /etc/raspimjpeg
if [ ! -e /var/www/${rpicamdir}/raspimjpeg ]; then
    sudo ln -s /etc/raspimjpeg /var/www/${rpicamdir}/raspimjpeg
fi

sudo usermod -a -G video www-data
if [ -e /var/www/${rpicamdir}/uconfig ]; then
    sudo chown www-data:www-data /var/www/${rpicamdir}/uconfig
fi

fn_motion
fn_autostart

if [ -e /var/www/${rpicamdir}/uconfig ]; then
    sudo chown www-data:www-data /var/www/${rpicamdir}/uconfig
fi

if [ -e /var/www/${rpicamdir}/schedule.php ]; then
    sudo rm /var/www/${rpicamdir}/schedule.php
fi

sudo sed -e "s/www/www${rpicamdirEsc}/g" www/schedule.php > www/schedule.php.1
sudo mv www/schedule.php.1 /var/www/${rpicamdir}/schedule.php
sudo chown www-data:www-data /var/www/${rpicamdir}/schedule.php

if [ $# -eq 0 ] || [ "$1" != "q" ]; then
    fn_reboot
fi

```

## *./Cam.sh*

```
#!/bin/bash

cd $(dirname $(readlink -f $0))

if [ $(dpkg-query -W -f='${Status}' "dialog" 2>/dev/null | grep -c "ok installed") -eq 0 ]; then
    sudo apt-get install -y dialog
fi

# Terminal colors
color_red="tput setaf 1"
color_green="tput setaf 2"
color_reset="tput sgr0"
RED=$(tput setaf 1)
GREEN=$(tput setaf 2)
NORMAL=$(tput sgr0)

ASK_TO_REBOOT=0
RASPICONFIG=/boot/config.txt
MAINSCRIPT="./RPi_Cam_Web_Interface_Installer.sh"

# Version stuff moved out functions as we need it more when one time.
versionfile="./www/config.php"
version=$(cat $versionfile | grep "'APP_VERSION'" | cut -d '"' -f4)
backtitle="Copyright (c) 2014, Silvan Melchior. RPi Cam $version"

get_config_var() {
    lua - "$1" "$2" <<EOF
    local key=assert(arg[1])
    local fn=assert(arg[2])
    local file=assert(io.open(fn))
    local found=false
    for line in file:lines() do
        local val = line:match("^%s"..key.."="(.)$")
        if (val ~= nil) then
            print(val)
            found=true
            break
        end
    end
    if not found then
        print()
    end
    EOF
}

set_config_var() {
    lua - "$1" "$2" "$3" <<EOF > "$3.bak"
    local key=assert(arg[1])
    local value=assert(arg[2])
    local fn=assert(arg[3])
    local file=assert(io.open(fn))
}
```

```

local made_change=false
for line in file:lines() do
    if line:match("^#?%s*"..key.."="..*$") then
        line=key.."="..value
        made_change=true
    end
    print(line)
end

if not made_change then
    print(key.."="..value)
end
EOF
mv "$3.bak" "$3"
}

do_finish() {
    if [ $ASK_TO_REBOOT -eq 1 ]; then
        dialog --yesno "Would you like to reboot now?" 5 35
        if [ $? -eq 0 ]; then # yes
            sync
            reboot
        elif [ $? -eq 1 ]; then # no
            exec sudo $MAINSRIPT
        fi
    fi
    exit 0
}

do_camera ()
{
    if [ ! -e /boot/start_x.elf ]; then
        dialog --msgbox "Your firmware appears to be out of date (no start_x.elf). Please update" 20 60
        exec sudo $MAINSRIPT
    fi
    dialog --title "Raspberry Pi camera message" \
        --backtitle "$backtitle" \
        --extra-button --extra-label Disable \
        --ok-label Enable \
        --yesno "Enable support for Raspberry Pi camera?" 5 48
    response=$?
    case $response in
        0) #echo "[Enable] key pressed."
            set_config_var start_x 1 $RASPICONFIG
            CUR_GPU_MEM=$(get_config_var gpu_mem $RASPICONFIG)
            if [ -z "$CUR_GPU_MEM" ] || [ "$CUR_GPU_MEM" -lt 128 ]; then
                set_config_var gpu_mem 128 $RASPICONFIG
            fi
            sed $RASPICONFIG -i -e "s/^startx/#startx/"
            sed $RASPICONFIG -i -e "s/^fixup_file/#fixup_file/"
            ASK_TO_REBOOT=1
            do_finish
            ;;
        1) #echo "[Cancel] key pressed."
            exec sudo $MAINSRIPT
            ;;
    esac
}

```

```

3) #echo "[Disable] key pressed."
set_config_var start_x 0 $RASPICONFIG
sed $RASPICONFIG -i -e "s/^startx/#startx/"
sed $RASPICONFIG -i -e "s/^start_file/#start_file/"
sed $RASPICONFIG -i -e "s/^fixup_file/#fixup_file/"
    ASK_TO_REBOOT=1
    do_finish
;;
255) echo "[ESC] key pressed."
    exec sudo $MAINSRIPT
;;
esac
}

do_camera
exit

```

## ./Start.sh

```

exec 5> start.txt
BASH_XTRACEFD="5"
set -x

cd $(dirname $(readlink -f $0))

source ./config.txt

fn_stop ()
{ # This is function stop
    sudo killall raspimjpeg 2>/dev/null
    sudo killall php 2>/dev/null
    sudo killall motion 2>/dev/null
}

#start operation
fn_stop
sudo mkdir -p /dev/shm/mjpeg
sudo chown www-data:www-data /dev/shm/mjpeg
sudo chmod 777 /dev/shm/mjpeg
sleep 1;sudo su -c 'raspimjpeg > /dev/null &' www-data
if [ -e /etc/debian_version ]; then
    sleep 1;sudo su -c "php /var/www/$rpicamdir/schedule.php > /dev/null &" www-data
else
    sleep 1;sudo su -c '/bin/bash' -c "php /var/www/$rpicamdir/schedule.php > /dev/null &" www-data
fi

```

## *./Stop.sh*

```
exec 5> stop.txt
BASH_XTRACEFD="5"
set -x

cd $(dirname $(readlink -f $0))

source ./config.txt

fn_stop ()
{ # This is function stop
  sudo killall raspimjpeg 2>/dev/null
  sudo killall php 2>/dev/null
  sudo killall motion 2>/dev/null
}

#stop operation
fn_stop
```