Imperial College London

EE YEAR 2 PROJECT FINAL REPORT

Drowsy Driving Detection System ----Group 6

Supervisor: Dr. Panagiota Tania STATHAKI
Group Member: Sajid Ali (01341393), Sheng Yu (01413225), Qiyu Xiong(01327233), Sergei Chirkunov(01374387), David Adeyemi(01364508), Joshua Afengbai(01353681), Angus Joshi(01374387), Tayyab Bhanadari(01334547)

CONTENT

I. Abstract

The aim of our project was to demonstrate and test a system which could detect whether a driver is drowsy by using an array of sensors (two in our case). The data from these sensors is then processed by comparing it to a data base or our model. We managed to get operational two sensors. One monitors the drivers steering patterns to detect sudden jerky movements which our research suggested was symptomatic of drowsiness. The other sensor is a camera setup which monitors the number of blinks and the duration of these blinks over a period of time, and is capable of operating in low light conditions, since the earlier and morning and night are the times when most incidents involving drowsiness occur. This was also found to be an accurate indication of drowsiness. We hoped that by combining the results from both sensors it would give an accurate picture of the state of the driver. The data gathered and processed from both sensors was accurate and given the cost of the items and the ubiquity of availability, we believe that the growing awareness of law makers and the general public to the risks associated with drowsy driving give this system commercial viability.

II. Introduction/Background

2.1 Background

The risks associated with driving under the influence are well known by law makers and the general public. What is less widely known (but increasingly changing) are the dangers of driving while drowsy. The National Highway Traffic Safety Administration in the USA estimates that drowsiness plays a factor in over 100,000 crashes, resulting in 6,550 deaths annually [1], and that this is becoming an increasingly common outcome as more people travel longer distance for work. The rise of online shopping may also be contributing to this phenomenon as there are more couriers on the road working longer hours with limited government regulation when it comes to working hours. The danger is that by the time drivers realise they are dosing off it may be too late to alert them and result in serious damage. Our proposal aims to prevent this by alerting the driver when they are falling asleep or losing concentration. We hope to create a low-cost system which can easily be implemented and imbedded within vehicles by using existing technologies and data processing to detect drowsiness.

2.2 Sub System Development

Our project focussed primarily on the detection of drowsy driving by employing sensors to gather different. We decided to have one sensor which would monitor the driving patterns of the driver, and another sensor which would monitor and attempt to find any physiological symptoms of drowsiness. Some of the other systems which we could have developed in this project include a system to alert the drive, embedding the system within vehicle, contacting emergency services in the event of a crash.

III. Design Criteria

Below are the design criteria which we created during our preliminary report, and the basis by which the success of the final solution will be evaluated.

Performance

Device needs to be able to detect whether the driver is drowsy, tired, or driving chaotically and inform them. There should be a low probability of registering a false positive, while being effective at detecting actual hazards.

Performance is important because without the ability to generate reliable results the system's effectiveness is dramatically reduced. Since the system is only expected to act in the rare occasion that the driver is dosing off, it is expected to have a high level of precision. Performance should be biased to giving a warning over not because an occasional false positive is preferred over not warning the driver when there is actual danger.

Cost

The total cost of this product for researching and prototyping should be below 200 pounds. In the future, the average cost of the product would be around 40 pounds.

One of the unique selling points of our system will be its low cost and ease of implementation. Therefore, it is crucial that production costs are kept to a minimum and as much hardware is re-used as possible.

Timescale

The general design of the product would be given by the end of the autumn term. The manufacturing would be carried out from the spring term. In the end, a demo and post display would be given.

Completing the design within the specified time may mean reducing the scope of the design and eliminating add-on features that we may have wanted to implement. We should anticipate challenges during the design and building phase and give ourselves enough time to solve and overcome them.

Maintenance

The user shouldn't need to maintain the system other than the initial setup. Maintenance should be part of general vehicle safety checks.

Having a low and easy maintenance of the system will attract consumers and manufacturers.

Ergonomics

To make sure our user is delighted to use product, our product must be comfortable to wear and must not pose as a distraction to the driver as this would undermine the whole purpose of this product. The part of the product measuring physical activity should be light so it would be *comfortable on the driver's wrist/other body part. The product should not be made with colours too bright, so it doesn't distract the driver. The part of the product that performs image processing shouldn't cover too much of the speedometer or the windshield.*

If the device is attached to the driver it must be comfortable and not inhibit them from driving. Moreover, any other elements located on the dashboard should not be distracting or obstruct the driver's line of sight. The primary purpose of the system is to improve driver safety, therefore good ergonomics and design should allow the device to seamlessly integrate into the driving experience.

Competition

There are some hand bands available on the market already. These hand bands are able to detect and record the heart rate while it could not compare and send feedback signal. These products are supposed to monitor person's health situation while our product is designed to monitor person's concentration through monitoring his heart rate.

Studying competitors' solutions will allow us to meet our time schedule because we will spend less time hitting dead ends. Since, the market for this type of device is quite new and still developing its important to see what existing manufacturers have done and try to improve upon it.

Size

The device must be small enough not to disrupt or distract the driver. Ideally it should be embedded in the dashboard or mirror.

For wearable safety devices, it's important to ensure they do not distract the user from the task at hand. It is therefore critical to make the device compact so that it doesn't affect the driving experience.

IV. High Level Design

4.1 Chaotic Steering

The chaotic steering monitor signal flow is summarised in figure X. It shows that first the data from the accelerometer on the steering wheel is used to find the rate of change of the angle of the wheel. This value is then compared with a pre-determined threshold and the result of this comparison is the output.

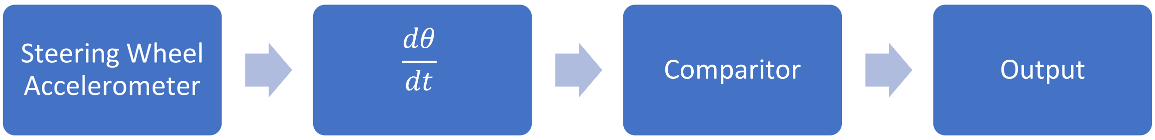


Figure 1 Chaotic steering sensor overview

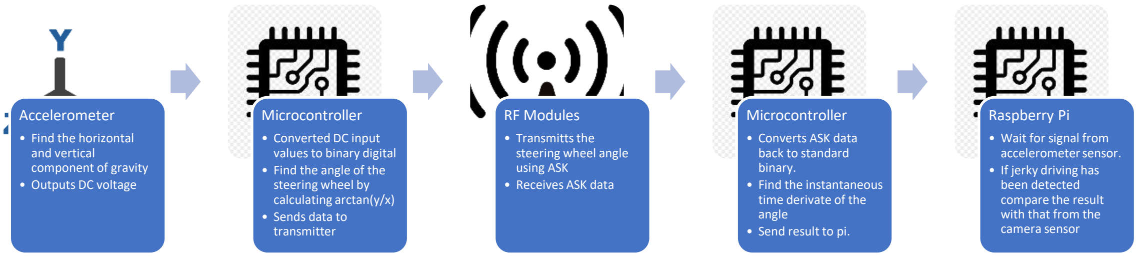
4.2 Blinking Detection

The blinking Detection module includes one NoIR (no infrared filter) camera, one Raspberry Pi, one BUCK converter, one phototransistor and several infrared LEDs. Images captured by the camera would be processed on the Raspberry Pi by a blinking detection algorithm written in python. Corresponding results would be displayed on screen. Infrared LEDs would be turned on to assistant the camera in the dark.

V. Detailed Design

5.1 Chaotic Steering

5.1.1 Overview



Accelerometer
- Find the horizontal and vertical component of gravity
- Outputs DC voltage

Microcontroller
- Converted DC input values to binary digital
- Find the angle of the steering wheel by calculating arctan(y/x)
- Sends data to transmitter

RF Modules
- Transmitts the steering wheel angle using ASK
- Receives ASK data

Microcontroller
- Converts ASK data back to standard binary.
- Find the instantaneous time derivate of the angle
- Send result to pi.

Raspberry Pi
- Wait for signal from accelerometer sensor.
- If jerky driving has been detected compare the result with that from the camera sensor

Figure 2 Chaotic steering sensor pipeline

5.1.2 Accelerometer Readings

In this system, the first task was to measure the x and y values for us to calculate the rate of change of the rotation of the steering wheel. We used an ADXL335 3-axis accelerometer with a minimum full-scale range of ± 3g. This accelerometer has a polysilicon surface micro machined sensor and signal conditioning circuitry to implement an open-loop acceleration measurement architecture. Additionally, it uses a single structure for sensing the X, Y, and Z axes which means that the three axes sense directions are highly orthogonal with little cross-axis sensitivity. The accelerometer outputs a DC voltage depending on the force through each of its axis.

We could have used a single axis on the plane of the steering wheels to determine the angle. But this posed a problems since the g-force through the axis is not a linear function of the angle. Instead it is a sinusoidal one which would mean we would have to scale the results. Moreover, the sinusoidal characteristic meant that some angle would have a large range of output values (steep part of slope), and other values would have a smaller number of discrete values to cover a similar range of angles. This would reduce the accuracy of the results for some ranges.

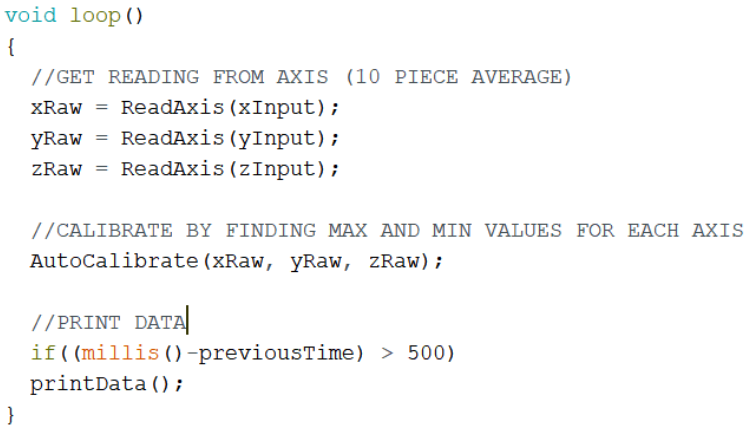
To solve this we adopted a method which would use both axis on the plane of the wheel. By finding the inverse tangent of the ratio of these values it would give us the angle. Since the two axis are offset from each other by 90 degrees, it means that when one is changing very rapidly, the other is changing very slowly. This gives a constant accuracy of the angle.

The accelerometer output a dc voltage which varies

$$V - \varepsilon \leq vout(\theta) \leq V + \varepsilon$$

When this voltage is converted to a digital number it will have an offset associated with it. Because we are finding the angle which requires us to calculate the inverse tangent of the x and y axis ratio, the value of x and y must be centred around 0. This means we need to remove the DC offset from the numeric results.

This was accomplished by finding the range of values corresponding to the voltage output of each axis.

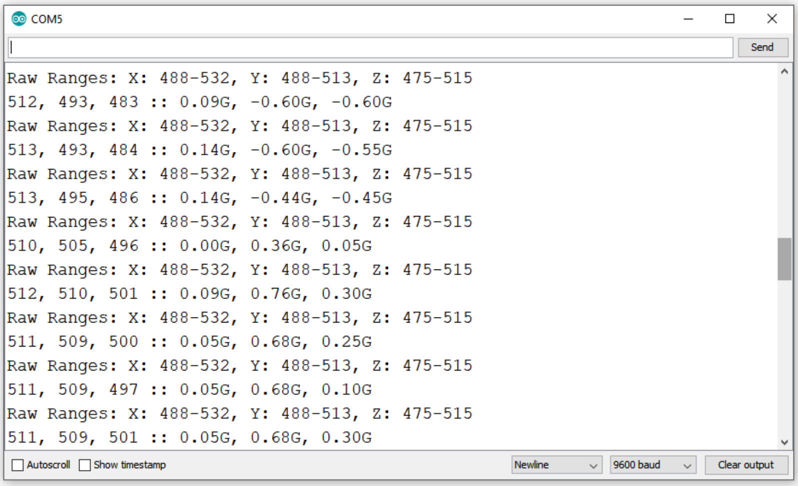


```
void loop()
{
  //GET READING FROM AXIS (10 PIECE AVERAGE)
  xRaw = ReadAxis(xInput);
  yRaw = ReadAxis(yInput);
  zRaw = ReadAxis(zInput);

  //CALIBRATE BY FINDING MAX AND MIN VALUES FOR EACH AXIS
  AutoCalibrate(xRaw, yRaw, zRaw);

  //PRINT DATA
  if((millis()-previousTime) > 500)
  printData();
}
```

Figure 3 Main loop of accelerometer calibration script

The code above shows the main loop of the program which attempts to find the range of values for each axis. This is done by changing the orientation of the sensor and allowing the program to keep and running account of the maximum and minimum values recorded for each axis.

The data gathered is show below. The final range is only accepted when it stops changing as the orientation changes. This means the maximum and minimum values have been found.



```
Raw Ranges: X: 488-532, Y: 488-513, Z: 475-515
512, 493, 483 :: 0.09G, -0.60G, -0.60G
Raw Ranges: X: 488-532, Y: 488-513, Z: 475-515
513, 493, 484 :: 0.14G, -0.60G, -0.55G
Raw Ranges: X: 488-532, Y: 488-513, Z: 475-515
513, 495, 486 :: 0.14G, -0.44G, -0.45G
Raw Ranges: X: 488-532, Y: 488-513, Z: 475-515
510, 505, 496 :: 0.00G, 0.36G, 0.05G
Raw Ranges: X: 488-532, Y: 488-513, Z: 475-515
512, 510, 501 :: 0.09G, 0.76G, 0.30G
Raw Ranges: X: 488-532, Y: 488-513, Z: 475-515
511, 509, 500 :: 0.05G, 0.68G, 0.25G
Raw Ranges: X: 488-532, Y: 488-513, Z: 475-515
511, 509, 497 :: 0.05G, 0.68G, 0.10G
Raw Ranges: X: 488-532, Y: 488-513, Z: 475-515
511, 509, 501 :: 0.05G, 0.68G, 0.30G
```

□ Autoscroll □ Show timestamp Newline 9600 baud

Figure 4 Serial terminal outputting axis ranges

Using the midpoint of the interval as the DC offset, I subtract this value from any reading I obtain from the sensors to ensure the values are centred around 0.

5.1.3 Data Transmission Protocol (Amplitude Shift Keying)

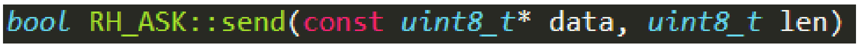
Amplitude shift keying (ASK) is a way of transmitting digital data by first converting it to an analogue form. To summarise, ASK transmits digital signals by multiplying the digital signal with the carrier signal. This has the effect of sending pulses of the carrier signal which correspond to 1s in the digital signal. Zeros are send as the gaps between the successive pulses. Demodulating an ASK signals is relatively easy because all it requires is an envelope detector.

We chose this method of transmission over other ones (such as FSK) because of the wide range of available RF modules which work with the method, and significant amount of existing open-source code and literature supporting the implementation of this protocol. Specifically we used the RadioHead library which supports ASK transmission for the Arduino microcontroller.

5.1.4 Transmitter

The figure below is a snippet from the code on the Arduino on the transmitter end. It includes the main loop function which executes every 100ms. It starts by reading the data from the analogue pins connected to the x and y axis of the accelerometer and subtracting form them the offset values discovered earlier. Next, the angle of the steering wheel is calculated by taking the inverse tangent of the ratio of the x and y values. The angles are then transformed to fit in the range of 0 to 360.

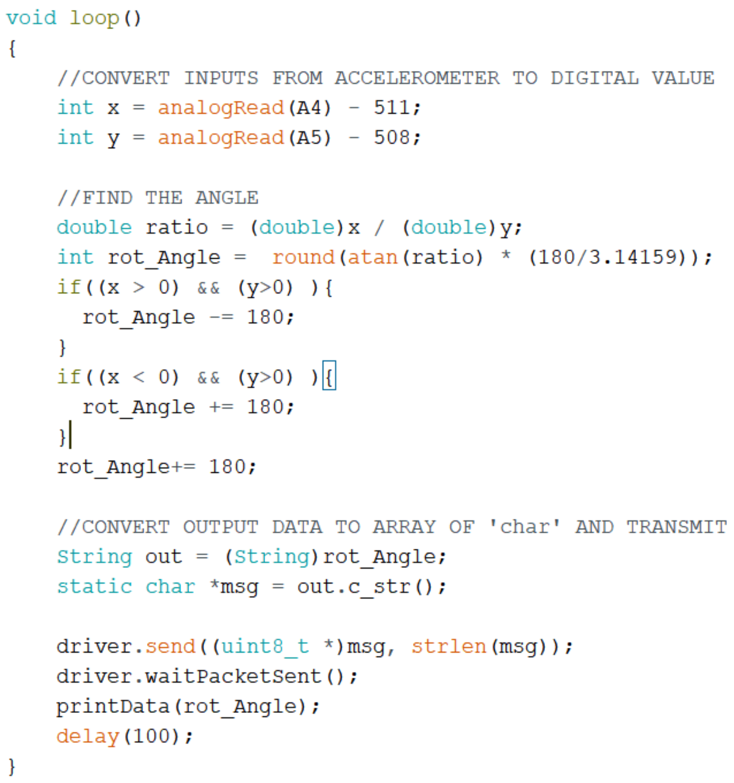
The library used to transmit the data requires that the data to be in byte form. This is accomplished by converting rot_Angle to an array of characters (i.e. a string). This works because the data type for a 'char' is 'uint8_t' which is an unsigned 8 bit integer (the definition of a byte). Finally the data is transmitted by calling the 'send()' method of the ASK Driver object. The full code for the sender function can be found in the appendix, but to summarise it works as follows.



```
bool RH_ASK::send(const uint8_t* data, uint8_t len)
```

Figure 5 Signature of ASK.send method

- Wait until the transmitter is available
- Encode the message length
- Encode the header
- Encode the message into 6 bit symbols (each byte is converted into 2 6-bit symbol)
- Transmit Data



```
void loop()
{
    //CONVERT INPUTS FROM ACCELEROMETER TO DIGITAL VALUE
    int x = analogRead(A4) - 511;
    int y = analogRead(A5) - 508;

    //FIND THE ANGLE
    double ratio = (double)x / (double)y;
    int rot_Angle =  round(atan(ratio) * (180/3.14159));
    if((x > 0) && (y>0) ){
      rot_Angle -= 180;
    }
    if((x < 0) && (y>0) ){
      rot_Angle += 180;
    }
    rot_Angle+= 180;

    //CONVERT OUTPUT DATA TO ARRAY OF 'char' AND TRANSMIT
    String out = (String)rot_Angle;
    static char *msg = out.c_str();

    driver.send((uint8_t *)msg, strlen(msg));
    driver.waitPacketSent();
    printData(rot_Angle);
    delay(100);
}
```

Figure 6 Main loop of transmitter script

The circuit diagram for the transmitter if show below in figure X. Both the RF transmitter and accelerometer are 5v devices which means they are easy to integrate with the Arduino. Moreover, the lack of a serial interface with the devices made them easier to use but may limit expandability in the future and make it harder to integrate them with other sensors.

The entire sensor is run off a 9v battery which supplies the Arduino with power, which intern feeds the peripheral components. The 9v batter was chosen because it is in the recommended voltage range to power the Arduino (7-12v), and is more robust and longer lasting than other lower powered alternatives such as coin batteries. Moreover, for our testing purposes the power supply should not be simulated as a limiting factor because in practice the sensor system would be directly connected to the vehicle and not require its own discrete power supply.

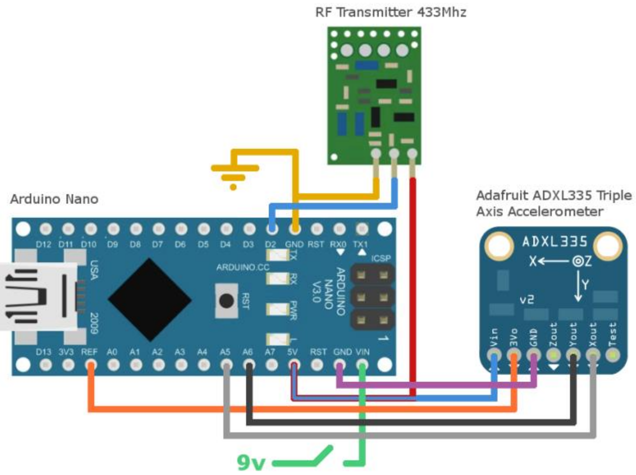
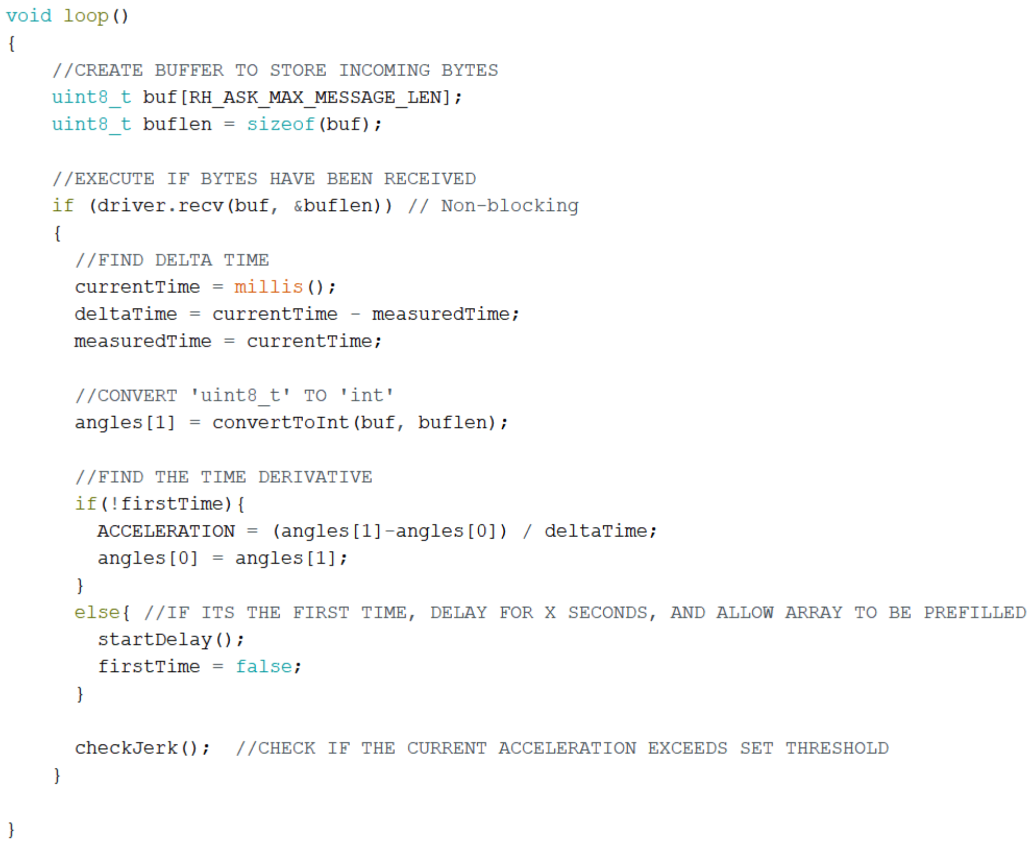

RF Transmitter 433Mhz
Arduino Nano
Adafruit ADXL335 Triple Axis Accelerometer
D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX1
D13 3V3 REF A0 A1 A2 A3 A4 A5 A6 A7 5V RST GND VIN
TX RX PWR L
ICSP
ARDUINO.CC
RST
ARDUINO NANO V3.0
USA 2009
ADXL335
X Z Y
v2
Vin 3Vo GND Zout Yout Xout Test
9v

Figure 7 Transmitter circuit diagram

5.1.5 Receiver

The main loop of the receiver Arduino is shown below. To begin with an array of unsigned bytes is created to store incoming data. Since the size of the data packets is unknown, the array is initialised to its maximum size to ensure all the data is captured. The size of the incoming data packets is stores in another byte which will inform the program how much of the previously declared array is filled. Next the ASK method recv() is called. This returns a Boolean depending on if data is received. If data is received the following procedure is implemented.

- Find the time elapsed since the last time data was received.
- Convert the incoming byte to int data types to allow for numerical transformations and comparisons.
- Add a delay if this is the first time the loop is running. This allows the program to settle and prevents spikes in the output data.
- Calculate the acceleration of the steering wheel by finding the difference between the received angle and the previous angle, an then divide by the time elapsed since the previous angle was received.
- Finally, check for jerky movement by comparing the magnitude of the acceleration with a pre-defined threshold. If it exceeds this threshold a warning is triggered.



```
void loop()
{
    //CREATE BUFFER TO STORE INCOMING BYTES
    uint8_t buf[RH_ASK_MAX_MESSAGE_LEN];
    uint8_t buflen = sizeof(buf);

    //EXECUTE IF BYTES HAVE BEEN RECEIVED
    if (driver.recv(buf, &buflen)) // Non-blocking
    {
      //FIND DELTA TIME
      currentTime = millis();
      deltaTime = currentTime - measuredTime;
      measuredTime = currentTime;

      //CONVERT 'uint8_t' TO 'int'
      angles[1] = convertToInt(buf, buflen);

      //FIND THE TIME DERIVATIVE
      if(!firstTime){
        ACCELERATION = (angles[1]-angles[0]) / deltaTime;
        angles[0] = angles[1];
      }
      else{ //IF ITS THE FIRST TIME, DELAY FOR X SECONDS, AND ALLOW ARRAY TO BE PREFILLED
        startDelay();
        firstTime = false;
      }

      checkJerk();  //CHECK IF THE CURRENT ACCELERATION EXCEEDS SET THRESHOLD
    }

}
```

Figure 8 Main loop of Receiver script

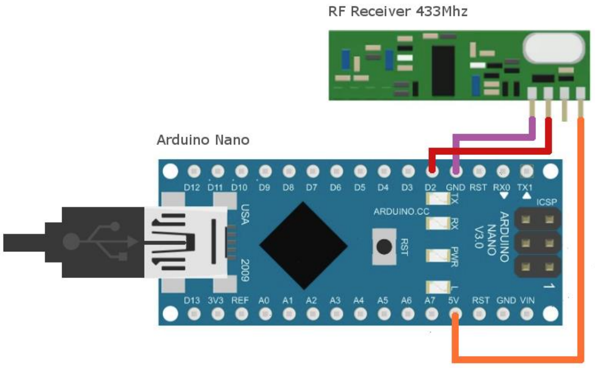
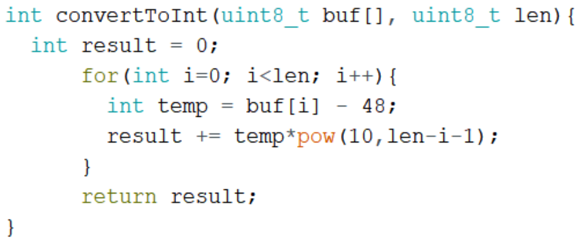

RF Receiver 433Mhz
Arduino Nano
D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX1
ICSP
USA
2009
ARDUINO.CC
RST
TX
RX
PWR
L
ARDUINO NANO V3.0
1
D13 3V3 REF A0 A1 A2 A3 A4 A5 A6 A7 5V RST GND VIN

Figure 9 Receiver circuit diagram

Since there were no functions to convert the incoming data stream from the data type many bytes to a single int, we created out own.

First we had to find the relationship between the incoming byte and the integer value they represented. We found that each byte was a character of the integer (i.e. 26 would come be a '2' and '6'). This meant that the original value could be recovered if each incoming byte is multiplied by descending powers of 10 from the largest power, and then summed together. The largest power of 10 was known since it was one less than the size of the incoming packets known from the recv() method.

$$result += buf[i] * 10^{(buflen-1)-i}$$



```
int convertToInt(uint8_t buf[], uint8_t len){
  int result = 0;
      for(int i=0; i<len; i++){
        int temp = buf[i] - 48;
        result += temp*pow(10,len-i-1);
      }
      return result;
}
```

Figure 10 Byte to Int conversion method

Determining the acceleration threshold for the warning to trigger required us to experiment and test different values. Similar to how we calibrated the senor, we simulated different driving conditions and monitored the acceleration when driving normally and when driving chaotically. From these experiments we were able to find that if the magnitude of the rate of change of the angle of the steering wheel is above 0.1 degrees per millisecond, or around 100 degrees per second, there is a high likelihood that the driver is exhibiting chaotic driving patterns.

5.1.6 Conclusion

The final sensors are displayed below. A switch was added to the transmitter circuit to reduce the battery drain and simulate the start/stop environment of a real vehicle. The sensor system beings functioning seconds after the power is switched on which means there is virtually no time when the vehicle would be on and the sensor is not operational.

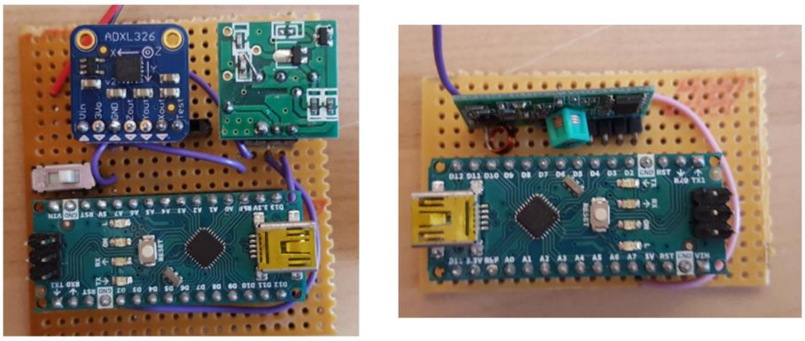


Figure 11 Left - Transmitter circuit; Right - Receiver circuit

5.2 Blinking Detection

5.2.1 Overview

In this subsystem, an algorithm written in python is applied to process images obtained by the camera. In addition, there is a hardware circuit built not only to provide the power supply for different components but also aiming to reduce the power dissipation.

5.2.2 Algorithm

There are two main tasks of this algorithm. Firstly, this algorithm is supposed to detect human faces from images and then detect blinking. Secondly, it should be able to count the number of blinks and trigger an alarm once the blinking counter is higher than a threshold value.

At the beginning, a NoIR camera is used to obtain images for the system. "NoIR" stands for "No Infrared Filter". Since this camera does not have an infrared filter, it could also detect infrared lights which is not visible to other normal cameras nor to human eyes. The reason of using a camera which could detect infrared lights is that according to a report from the U.S. National Highway Traffic Safety Administration (NHTSA), drowsy driving is highly likely to happen at late night which is the body's natural sleep period [1]. However, in a dark environment, a normal camera would not be able to capture images which are clear enough for facial detection. Furthermore, it is true that people always turn off lights inside their car while driving at night to see the outside more clearly; therefore, it is quite dangerous to turn on a light inside the cabin to assist the camera while driving, especially at night. However, instead of using visible lights, using infrared light could be practical because it cannot be seen by human eyes, but it could be detected by cameras without infrared light filter. In this system, a phototransistor would be used as a switch to control six infrared LEDs located around the camera. If the background environment is bright, the transistor is turned off and LEDs would not work. When the background environment is dark, either at night or in a tunnel, the transistor would turn on and infrared LEDs would send infrared lights to help the NoIR camera to capture images. The figure below is obtained by the NoIR camera in a dark environment with infrared LEDs turning on.

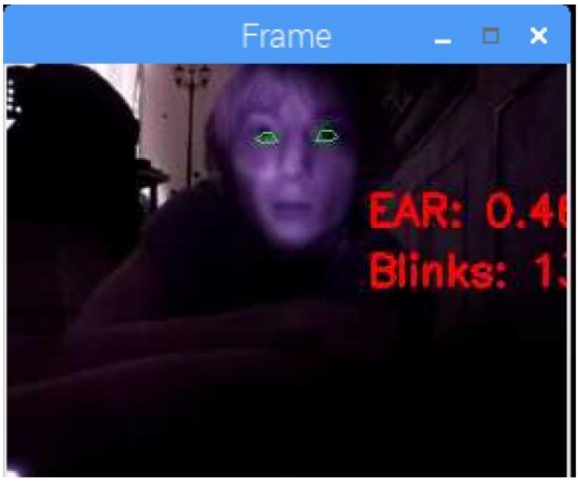
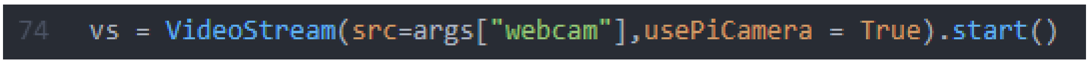


Figure 12 NoIR camera images in the dark

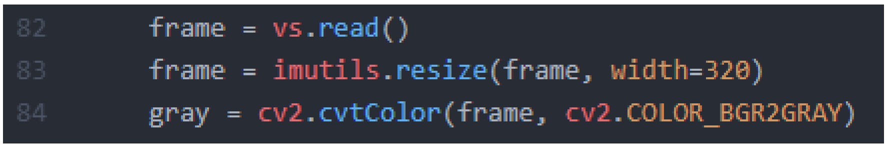
Once images are ready, a drowsiness detection code [2] starts to work (code attached as Appendix A). We firstly introduce imutils which is a series of convenient functions to make basic image processing functions such as rotation, resizing and detecting edges much easier with OpenCV. From imutils, a function named VideoStream is called. This function could use images obtained by the Pi camera to build a video stream for further processes.



```
vs = VideoStream(src=args["webcam"],usePiCamera = True).start()
```

Figure 13 Call VideoStream function

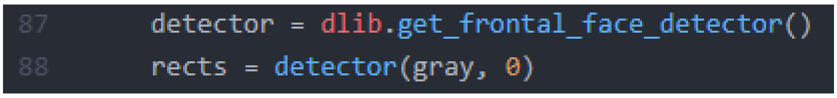
Next, we call OpenCV which is a library of programming functions mainly aimed at real time computer vision [2] to adjust images. We set the frame width to be 320. The frame size is limited by the hardware of the Raspberry Pi. In this project, the Raspberry Pi we used is a Pi 3 Model B which is the earliest model of the third-generation Raspberry Pi. It has a Quad Core 1.2GHz Broadcom BCM2837 64bit CPU and a 1GB RAM [4]. If the frame size it too big, it would take quite long time for the Pi to process each frame, which would cause lags. However, human blinking is very fast, a blinking might be missed due to lags of the video stream. Therefore, we set the frame size small to avoid lags. In addition, we also convert original RGB images to grayscale images. One of the most significant reasons for using grayscale images rather than colour images is that using grayscale could reduce the complexity of the code. Take the RGB colour image as an example, if each Red, Green and Blue take up 8 bits then the combination of an RGB image would take up 24 bits. The 24 bits represent the colour of a pixel in the colour image. For a grayscale image, it only uses 8 bits value to represent its luminance. The conversion of a colour image into a grayscale image is converting the RGB value which is 24 bits into the grayscale value which is only 8 bits [5]. Therefore, we convert colour images into grayscale images for processes.



```
        frame = vs.read()
        frame = imutils.resize(frame, width=320)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

Figure 14 Call OpenCV functions to adjust size and convert to grayscale

Then, we detect faces in the grayscale frame.



```
        detector = dlib.get_frontal_face_detector()
        rects = detector(gray, 0)
```

Figure 15 Detect faces

If a face is detected, the fourth step is to apply facial landmarks and calculate Eye Aspect Ratio (EAR) values. The facial landmark detector is implemented inside the dlib library. The detector would apply 68 landmarks with x and y coordinates to specific facial structures including eyes, nose and mouth. Then facial landmark coordinates are converted to a data array by a Python library called NumPy.

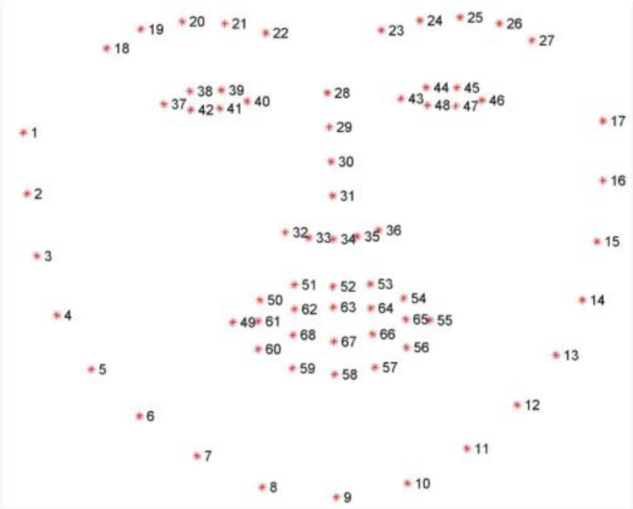


Figure 16 Visualizing 68 facial coordinate points

Since we are interested in eye blinking, we only focus on landmarks which represent eye features. There are twelve landmarks to outline both two eyes and each eye is surrounded by six landmarks. We would use those landmarks to calculate EAR values for both left and right eyes and take the average value.

EAR stands for Eye Aspect Ratio, which would characterize the eye opening in each frame [6]. The photo below illustrates one eye with its six landmarks around. EAR is defined by a function of those six landmarks:

$$\mathrm{EAR} = \frac{|P_2 - P_6| + |P_3 - P_5|}{2 * |P_4 - P_1|}$$

where P_1, P_2......$P_6$are coordinates of six landmarks around the eye. $|P_2 - P_6|$, $|P_3 - P_5|$ and $|P_4 - P_1|$ calculate the distance between three pairs, respectively.

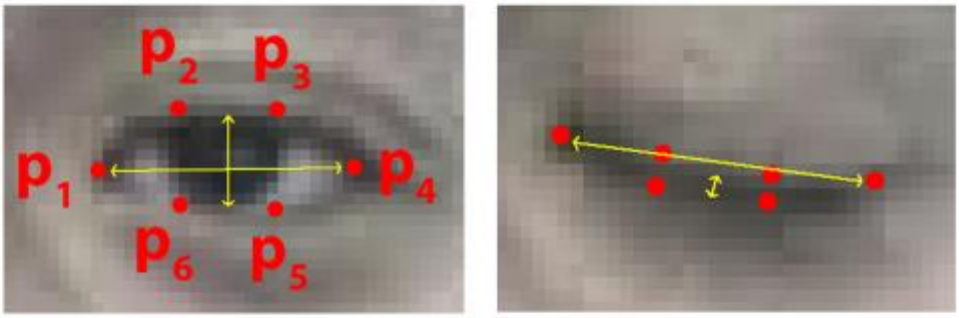


Figure 17 Open and Close Eyes with landmarks [6]

When the eye is closed, the distance between P_2 and P_6 would approach to zero. So is the distance between P_3 and P_5. This means the numerator of this equation would close to zero. On the other hand, the denominator of the equation, which is the distance between the Landmark 4 and the Landmark 1 would remain almost unchanged. Therefore, the value of this equation would decrease rapidly when eye is closed, which is illustrated in graph below.

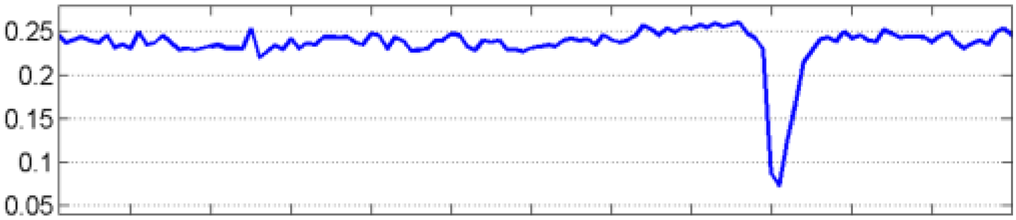


Figure 18 EAR plot for several frames. A single blink is displayed [6]

From the plot above, when the eye is open, the EAR value is around 0.25. In fact, we notice that the EAR values would be different for different eyes. For one of our group members, the EAR values are around 0.4 when his eyes are open and drop to 0.2 when he closes his eyes. Therefore, one of our user interfaces is that this product could be able to adjust EAR threshold for different users.

Real time EAR values as well as blinking numbers are displayed on the screen for testing and reference purposes.

We set the EAR Threshold to be 0.3. It means that once the EAR value is lower than 0.3, the system would record that eyes are closed for this this frame and COUNTER which is another parameter to trigger the alarm would be add one to itself. If the EAR value from the next frame is higher than the EAR Threshold, the COUNTER would reset to zero. This suggests that the previous eyes closing is a part of normal blinking. However, if the EAR value calculated from next frame is also smaller than the EAR Threshold, the COUNTER would keep adding. In this case, it indicates that for two consecutive frames, eyes are keeping closed, which is suspected of drowsy driving. The COUNTER value would suggest how long eyes have closed.

In order to avoid fake alarms, a threshold called COUNTER Threshold is introduced. For example, if the COUNTER Threshold is 15, it means only when 15 consecutive frames all give EAR values lower than the EAR Threshold will the Alarm be triggered.

In the end, once the alarm is triggered, a function from playsound library would be called to play an alarm sound and an alarm message would appear on the screen to remind the driver of not drowsy driving.

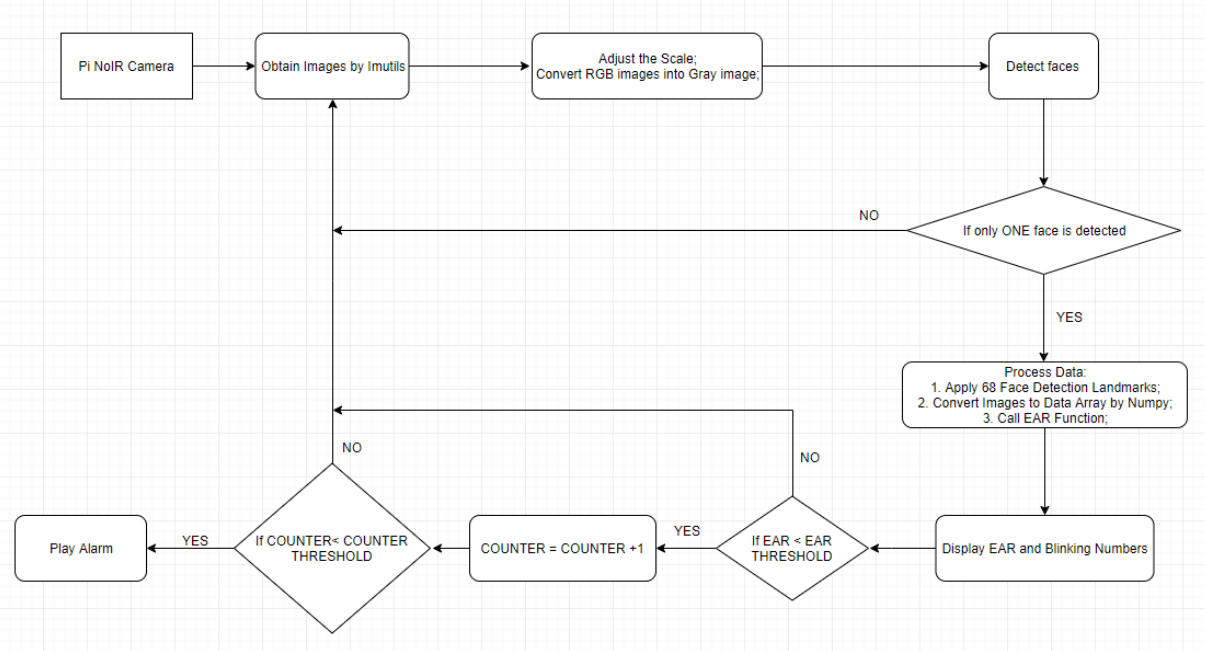

Pi NoIR Camera
Obtain Images by Imutils
Adjust the Scale;
Convert RGB images into Gray image;
Detect faces
NO
If only ONE face is detected
YES
Process Data:
1. Apply 68 Face Detection Landmarks;
2. Convert Images to Data Array by Numpy;
3. Call EAR Function;
NO
NO
Play Alarm
YES
If COUNTER< COUNTER THRESHOLD
COUNTER = COUNTER +1
YES
If EAR < EAR THRESHOLD
Display EAR and Blinking Numbers

Figure 19 Flow Chart of Blinking Detection Algorithm

5.2.3 Circuit Implementation

Another feature of our design is infrared lightning. We improved usability of our camera device by supporting it with 6 high intensity infrared LEDs. They are not distracting the driver and very compact. The only drawback they have is relatively high power consumption in comparison with Raspberry Pi 3. Unfortunately, for our first prototype we also experienced heating troubles. The voltage drop across LEDS is between 1.0-1.4V and the use of 5V power supply would be a complete waste of energy and we needed to reduce it. A simple potential divider was not quite a way out since we expected a current flow of around 1A. Therefore, we decided to design a little buck converter and we used MC34063 to do this. Carefully studying the data sheet and implementing example circuits we manufactured our own power control module. It has a maximum voltage input of 50V and maximum current of 1.5A. The key equations are:

$1.25(1+\frac{R_1}{R_2})$, $I_{pk} = \frac{300mV}{R_3}$ where 300mV is a current control voltage, $I_{pk} = 2I_{out}$

According to these equations we conclude that our output is independent of input, our integrated circuit has a build-in current limit and suitable current output to power LEDs.

The values of resistors are $R_1 = R_2 = 1k\Omega, R_3 = 0.33\Omega$

Our further improvement was a photoswitch. When sunlight is incident on the sensor it creates dark current through the base resistor which is increasing the voltage at the base and turning transistor into saturation. Later we replace our BC947 by ON Semi TIP31AG NPN Transistor, 3 A, 60 V. At this point we finished modifying the circuit. We still had some heating issues in the long run. The reason for it is that the IC was performing at its maximum, the inductor and the capacitors were taken from the EELab and not ordered specifically for the project. In future this can be significantly improved by replacing our existing components by high power ones.

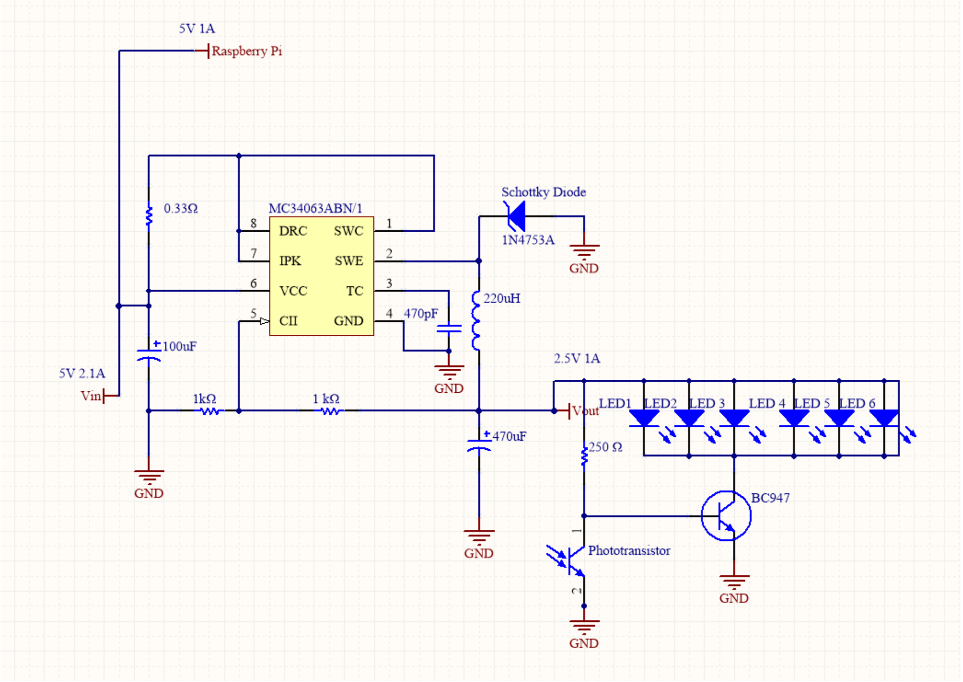


Figure 20 Circuit Diagram for Blinking Detection System

5.2.4 Raspberry Pi Implementation

Face detection was implemented using Raspberry Pi there is a list of recommendations how to operate raspberry and keep high performance that we outlined during this project. We used a clear setup with GUI and step by step improved to what we have now. In the end, our raspberry used only relevant libraries and focused its computational power for Image Analysis and VNC server only.

This is a list of the most import command to create a comfortable environment:

sudo raspi-config Advanced options->Expand Filesystem Interface Options-> Enable SSH, VNC, Serial	First steps to communicate
sudo raspi-config Advanced options->swap area (Increase to 2048)	In order to install OpenCV we need more memory
Cmake -j4	Use before building the OpenCV library to use all 4 cores
dpkg-query -Wf '${Installed-Size}\t${Package}\n' | sort -n Sudo apt-get purge -y packagename	Scan for packages and delete them
sudo mount --bind / /mnt sudo ncdu -x /mnt	Scan filesystem

country=UK
ctrl_interface=DIR=/var/run/wpa_supplicant
GROUP=netdev
update_config=1
network={
ssid="Imperial"
	Network configuration setup for imperial wifi
	Create the following file and save the code in the boot drive when sd card is plugged to your device.
	wpa_supplicant.conf
	
	Also create SSH.txt to enable ssh communication

psk=""
}	

https://www.raspberrypi.org/documentation/configuration/wireless/access-point.md	Setting up an access point for further Communication via VNC server and SSH
https://github.com/jrosebr1/imutils/blob/master/imutils/video/videostream.py https://github.com/jrosebr1/imutils/blob/master/imutils/video/pivideostream.py	Enable PiCamera libraries and NOIR camera for Imutils Library
https://lifehacker.com/how-to-clone-your-raspberry-pi-sd-card-for-super-easy-r-1261113524	Clone sd card to save time in the future
https://raspberrypi.stackexchange.com/questions/7088/playing-audio-files-with-python	Installing pygame for sound outputut
https://www.raspberrypi.org/forums/viewtopic.php?f=63&t=199672	Create autorun to fix latest update with dnsmasq and hostapd services
sudo systemctl unmask dnsmasq sudo systemctl unmask hostapd https://www.digitalocean.com/community/tutorials/how-to-use-systemctl-to-manage-systemd-services-and-units	Enable the following services for the correct operation of the Access point

Table 1 Important Commands

5.3 Embedded Design

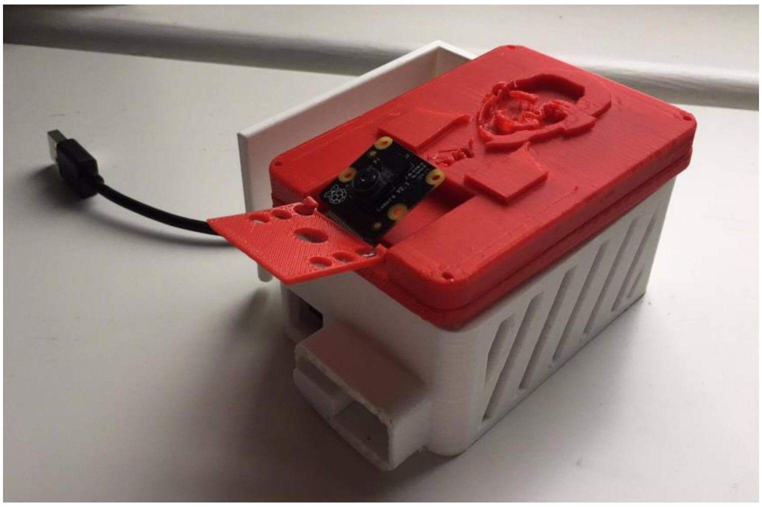
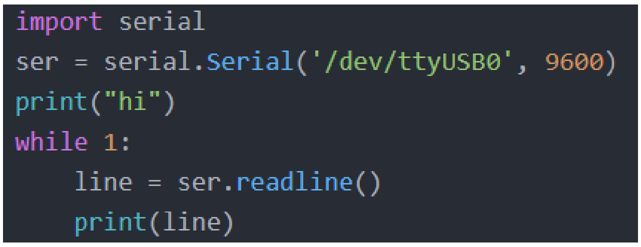


Figure 21 Prototype of the case

In the end, both sensors were embedded inside a PLA plastic case (details of the design in Appendix). We used serial Interface to communicate between Arduino Nano and Raspberry PI.



```
import serial
ser = serial.Serial('/dev/ttyUSB0', 9600)
print("hi")
while 1:
    line = ser.readline()
    print(line)
```

The idea is simple however a conversion of the input to an integer was necessary. For serial interface accelerometer sent '1' if chaotic motion was detected and 0 otherwise. Data for accelerometer was synchronised with the frame processing and therefore the accuracy of the sensor was based on the frequency of threading (in other words one accelerometer reading per one frame). Therefore, synchronised frequency was defined by the number of frames per second (defined by pure performance of OpenCV and Dlib algorithms). Figure 20 shows how the data is evaluated.

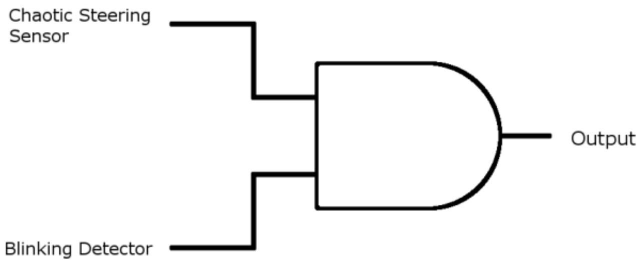


Figure 22 Combined sensor output

VI. Project Management

6.1 Roles and Responsibility

In order to collaborate as a team, each group member was assigned to a specific role:

Roles	Members	Responsibilities
Project leader	Sajid Ali	Coordinating the team to handle the tasks and assigning tasks to individuals.
Secretary	Sheng Yu	Recording important outcomes of group decision and reminding members with internal and external deadlines.
treasurer	Qiyu Xiong	Ordering electrical components online for prototyping.

Table 2 Group member roles

6.2 Technical Groups

To achieve higher efficiency and enable every member to focus on specific fields, the whole group is divided into two sub-groups focusing on two different technical aspects, this is shown below:

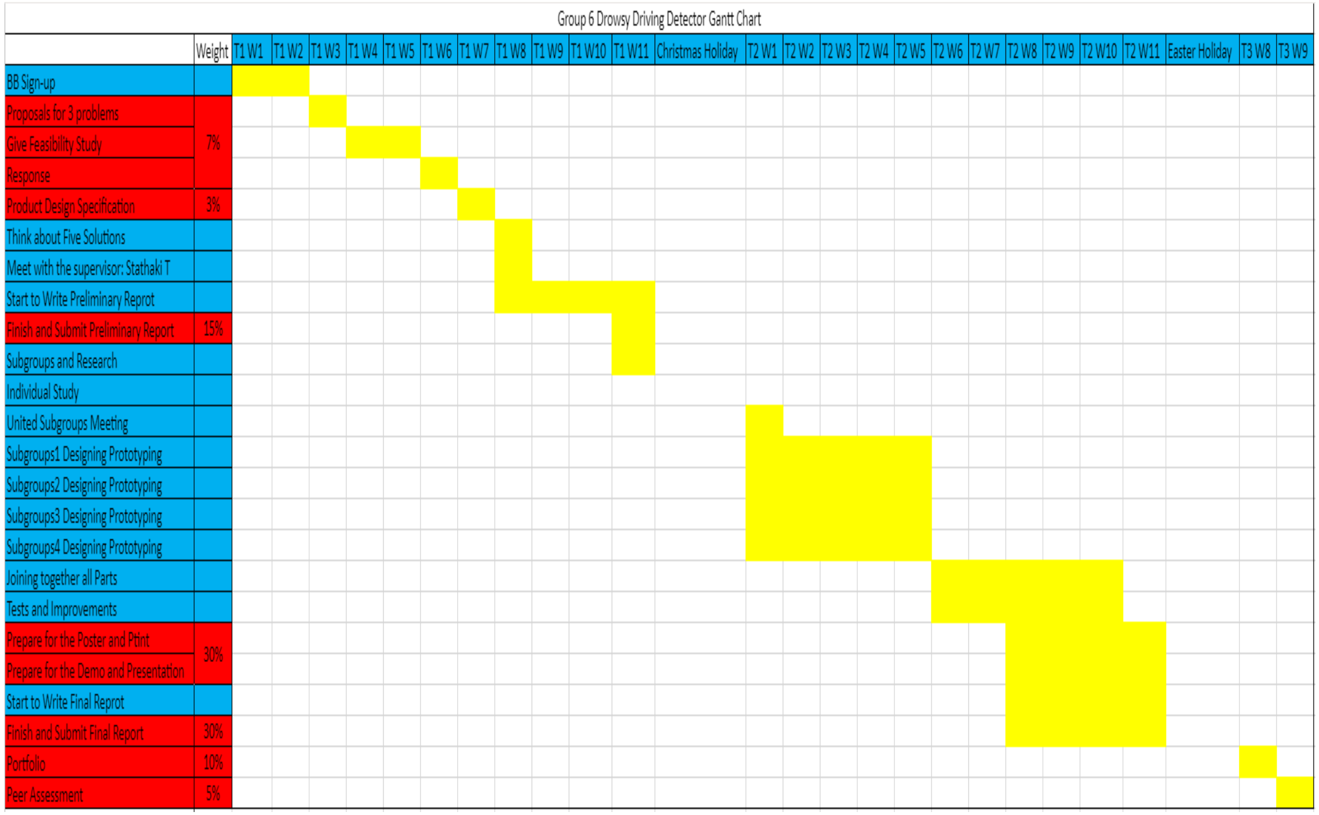
Sub-group	Members
Steering accelerometer	Sajid Ali David Adeyemi Joshua Afengbai Tayyab Bhandari Angus Joshi
Blinking detection	Sheng Yu Sergei Chirkunov Qiyu Xiong

Table 3 Sub-groups

Each group would work separately and complete the tasks given before every Monday meeting at 1 pm. In this meeting, members have opportunities to report their progress and outcomes. This meeting is also an important time for two sub-groups to know things in the other group and coordinate their work. Additionally, new works would be assigned to two sub-groups during this meeting, members in each sub-group will know their tasks in the afterward week.

6.3 Gantt Chart

The Gantt Chart was strictly followed by the team which is shown below:



Group 6 Drowsy Driving Detector Gantt Chart

	Weight	T1 W1	T1 W2	T1 W3	T1 W4	T1 W5	T1 W6	T1 W7	T1 W8	T1 W9	T1 W10	T1 W11	Christmas Holiday	T2 W1	T2 W2	T2 W3	T2 W4	T2 W5	T2 W6	T2 W7	T2 W8	T2 W9	T2 W10	T2 W11	Easter Holiday	T3 W8	T3 W9
BB Sign-up		🟨	🟨																								
Proposals for 3 problems	7%			🟨																							
Give Feasibility Study					🟨	🟨																					
Response							🟨																				
Product Design Specification	3%							🟨																			
Think about Five Solutions									🟨																		
Meet with the supervisor: Stathaki T									🟨																		
Start to Write Preliminary Reprot									🟨	🟨	🟨	🟨															
Finish and Submit Preliminary Report	15%											🟨															
Subgroups and Research												🟨															
Individual Study																											
United Subgroups Meeting														🟨													
Subgroups1 Designing Prototyping														🟨	🟨	🟨	🟨	🟨									
Subgroups2 Designing Prototyping														🟨	🟨	🟨	🟨	🟨									
Subgroups3 Designing Prototyping														🟨	🟨	🟨	🟨	🟨									
Subgroups4 Designing Prototyping														🟨	🟨	🟨	🟨	🟨									
Joining together all Parts																			🟨	🟨	🟨	🟨	🟨				
Tests and Improvements																			🟨	🟨	🟨	🟨	🟨				
Prepare for the Poster and Ptint	30%																				🟨	🟨	🟨	🟨			
Prepare for the Demo and Presentation																					🟨	🟨	🟨	🟨			
Start to Write Final Reprot																					🟨	🟨	🟨	🟨			
Finish and Submit Final Report	30%																				🟨	🟨	🟨	🟨			
Portfolio	10%																									🟨	
Peer Assessment	5%																										🟨

All tasks up to this point in the Gantt Chart is completed by us. As a result, there are only a few tasks we need to finish in next term:

1) To complete the assembly of final product.
2) To test the performance of our device under real driving condition. Try to test our device on a real car and ask the driver to behaviour differently to test its performance.
3) To make any adjustments and improvements based on its performance. The main point we focus on would be the comfortableness of user and whether it hinders real driving.

6.4 Critical Analysis

Part of working on a project is problem solving, especially those problems which were not anticipated. During the design and prototyping phase of development we encountered many hurdles which required us to adapt our plans and redesign parts of the system which we realised would no longer function. One of the first problems was to do with the computational limitation of the raspberry pi microcontroller. Initially we were planning on all the data processing happening on the pi, since it made sense to have it happen in a single central place where the output result could be easily obtained from. But we quickly realised that the computation involved in the image processing system might not leave enough space for the steering sensor computation. We quickly decided that it would be best if the processing for the steering sensor was to be done on the Arduino micro-controllers instead. This turned out to be a viable solution since all the Arduino were doing up until this point was transmitting and receiving data, which left plenty of room for additional processes. We split the computation across both processors which ended up giving us faster results and reduced the data being transmitted since converting the raw axis values of the accelerometers to angles was done on the transmitter Arduino thereby requiring us to send only a single integer across the RF channel as opposed to 2 integers previously.

Another issue we encountered was the unexpected breakdown of components. Many of the chips we purchased had different voltage ratings and poor soldering and incorrect circuit designs sometimes resulted in us accidently breaking components by sending too much current through them or exposing them to soldering heat for too long. To prevent this delaying production we made sure to order multiple sets of each components and implemented a double check policy in which each circuit would be checked by another group member before any permanent damage could be done. This especially helped during the later parts of the project when the system become more complex and easier to malfunction.

With our group consisting of 8 members working independently on their respective tasks it became crucial to maintain regular communication with all group member to ensure that we were meeting deadlines and staying on schedule. Sometimes communication broke down or responses came in too late which slowed production and produced error which in the end could not be remedied. One instance of this was a miscommunication on the dimensions of the accelerometer sensor housing which resulting in a housing which was too small to fit the sensor setup. Since the print took over 24 hours we were unable to re-print it with the correct dimensions, but luckily it was possible to make small adjustments to it which means the sensor would still be operational.

6.5 Budget Control

The budget is limited to £200. Because of this, it is very important to keep recording every component we buy and use the budget on necessary aspect. Below is the budget:

Module	Component	Specification	Quantity	Price per unit
Steering	Accelerometer	Adafruit ADXL335 5V ready triple-axis accelerometer (+-3g analogue out)	1	11.41
	ADC	MCP3008 4 Channel/8 Channel 10-Bit A/D Converters with SPI Serial Interface	4	2.15
	RF transceiver	RF Solutions SMARTALPHA-433 RF Transceiver Module 433 MHz	2	17.04
	Arduino Nano	Arduino Nano 3.0 Board with ATmega328	2	19
Blinking detection	NoIR Camera	Raspberry Pi NoIR Camera Board 8 Megapixel Version 2 1080p	1	20.49
	IR LED	Everlight IR333 IR LED Emitter 940nm 20° 5mm Radial	10	0.072
	NPN transistor	ON Semi TIP31AG NPN Transistor, 3 A, 60 V, 3-Pin TO-220AB	10	0.344
	Raspberry Pi	Raspberry Pi 3 Model B	1	Provided by member
	Diode	Vishay 50V 1A, Diode, 2-Pin DO-204AL 1N4001-E3/54	10	0.2

Total cost: £118.74

Budget remained: £81.26

VII. Future Work

1.) Currently the sensor system outputs a danger warning if both sensors detect a warning. A more sophisticated and robust method would monitor driving patterns over a period of time to detect long changes in alertness. This way the program would find out how alert the driver is and predict if the driver is exhibiting patterns of becoming drowsy in the future. This methods would increase safety because the driver would be reminded to take a break before they are driving dangerously. Also, making the sensors work better together would increase reliability.

2.) Increasing the number of sensors would further increase the reliability of the system and reduce the risk associated with a single sensor malfunctioning.

3.) In our project we focuses exclusively on detection of drowsy driving. But once it has been found that the driver is not safe, it is equally important to inform the driver somehow to stop them driving without causing stress and increasing the danger level.

4.) If we wanted to keep to the algorithms we created, it is also crucial that the system is dynamic enough to adapt to the different driving patterns and blinking rates of a wide range of drivers in different vehicles.

5.) The current system is more about data processing than data collection. Using more sophisticated algorithms such as machine learning and neural networks will enable us to get better results. Ideally, we want to mesh the accuracy of neural networks with the ease of implementation and theoretical frame work of mathematical models. This would reduce the need for lots of input data for the algorithm.

6.) The system could also form a good way of getting data which could later be used to create neural networks in the future by providing the raw data required to create such a system.

VIII. Conclusion

In the end, we a satisfied with the progress we made and are optimistic about the commercial viability of a more developed system. We managed to obtain reliable and consistent data from both sensors while sticking to within our allocated budget. Working as part of a large team can always be a challenge, but everyone contributed to the final outcome and worked well together, especially with members of their sub-group.

IX. References

[1] U.S. Department of Transportation, "Drowsy Driving", *NHTSA*, April 3, 2018. [Online]. Available: https://www.nhtsa.gov/risky-driving/drowsy-driving [Accessed: Mar. 22, 2019].

[2] Sleep Education, "Drowsy Driving", 2018. [Online]. Available: http://sleepeducation.org/sleep-topics/drowsy-driving [Accessed: Mar. 22, 2019].

[3] A. Rosebrock, "Eye blink detection with OpenCV, Python, and dlib", April 24, 2017. [Online]. Available: https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/ [Accessed: Mar. 22, 2019].

[4] K. Pulli, A. Baksheev, K. Kornyakov, V. Eruhimov, "Realtime Computer Vision with OpenCV", *Queue*, Volume 10, Issue 4, p.40, April 2012. [Online]. Available: https://dl.acm.org/citation.cfm?id=2206309 [Accessed: Mar. 22, 2019].

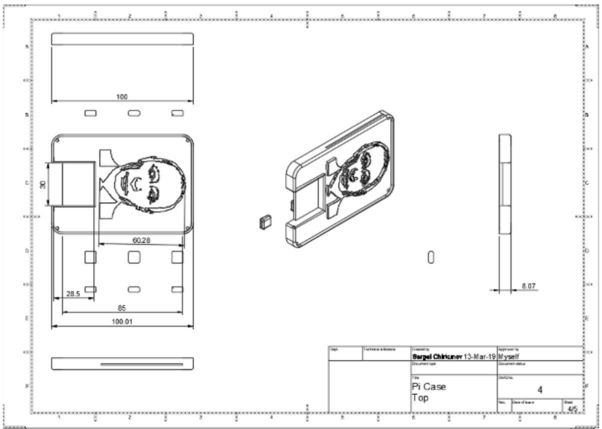
[5] RaspberryPi, "Rapberry Pi 3 Model B Specifications", [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ [Accessed: Mar. 22, 2019].

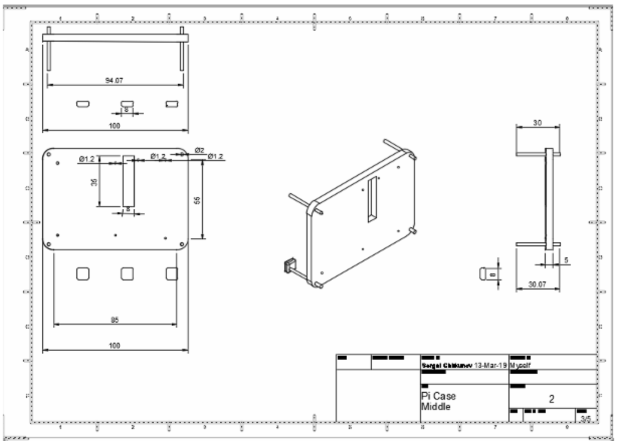
[6] C. Saravanan, "Color Image to Grayscale Image Conversion", *2010 Second International Conference on Computer Engineering and Applications,* March 2010. DOI: 10.1109/ICCEA.2010.192, Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5445596 [Accessed: Mar. 22, 2019].

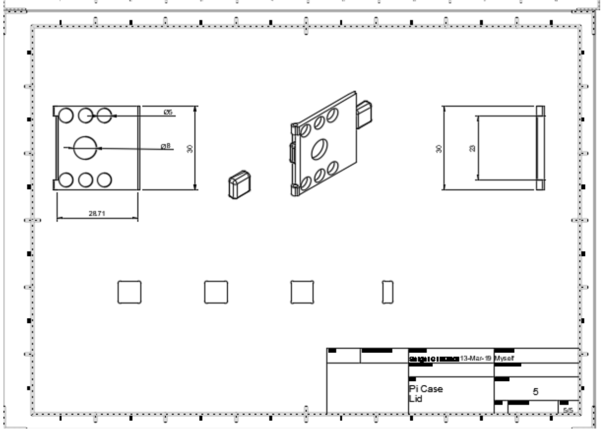
[7] T. Soukupova, J. Cech, "Real-Time Eye Blink Deection using Facial Landmarks", *21st Computer Vision Winter Workshop*, Feb. 2016. [Online]. Available: http://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf [Accessed: Mar. 22, 2019]

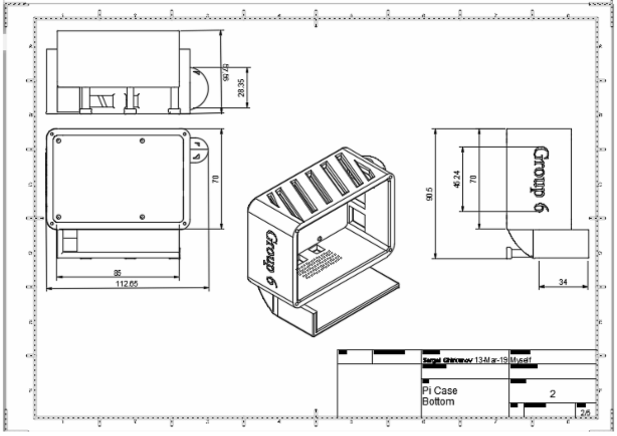
X. Appendix

1. 3D printed sensor housing designs

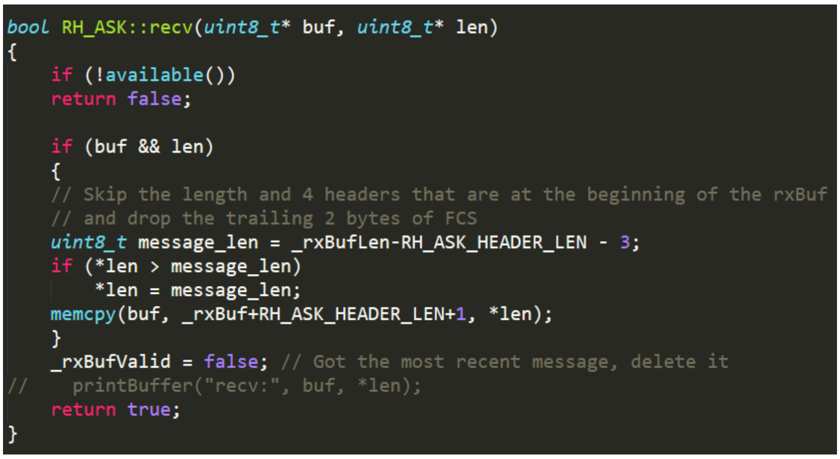




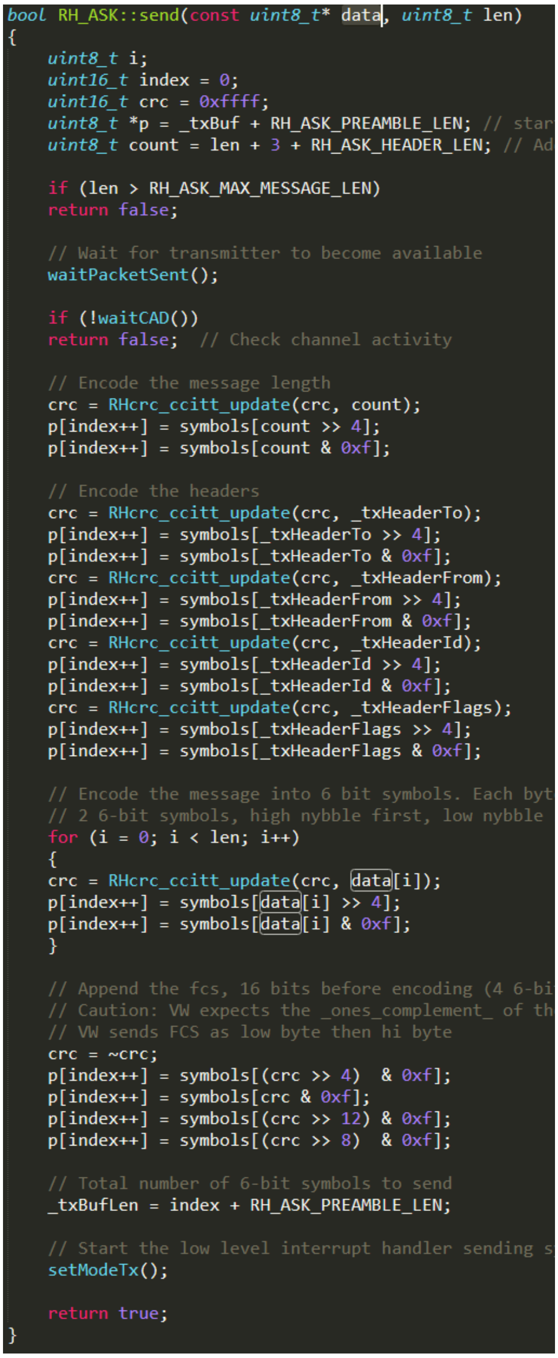




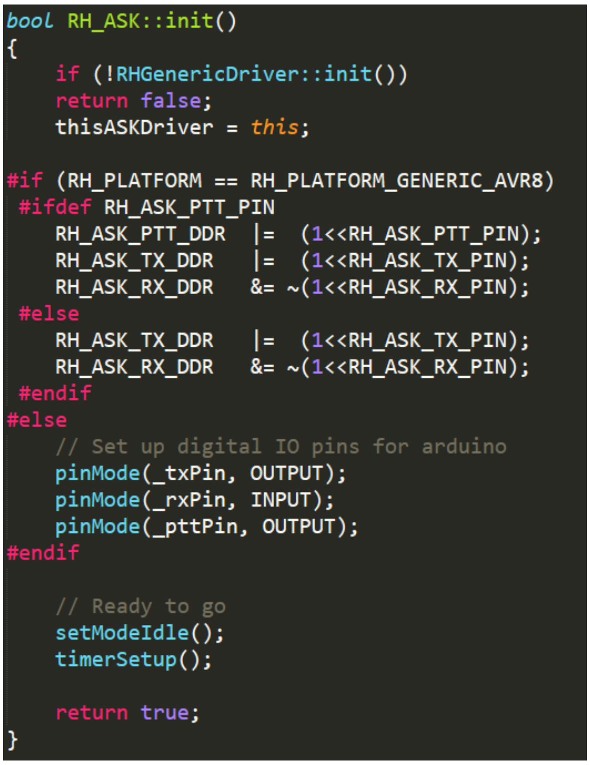
2. RadioHead ASK Methods



```
bool RH_ASK::recv(uint8_t* buf, uint8_t* len)
{
    if (!available())
    return false;

    if (buf && len)
    {
    // Skip the length and 4 headers that are at the beginning of the rxBuf
    // and drop the trailing 2 bytes of FCS
    uint8_t message_len = _rxBufLen-RH_ASK_HEADER_LEN - 3;
    if (*len > message_len)
        *len = message_len;
    memcpy(buf, _rxBuf+RH_ASK_HEADER_LEN+1, *len);
    }
    _rxBufValid = false; // Got the most recent message, delete it
//    printBuffer("recv:", buf, *len);
    return true;
}
```

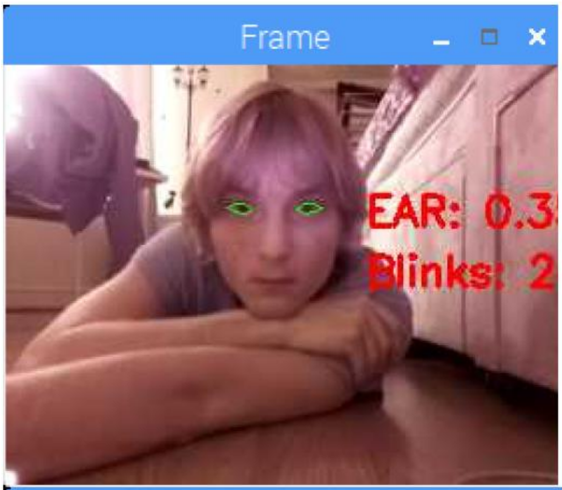


```
bool RH_ASK::send(const uint8_t* data, uint8_t len)
{
    uint8_t i;
    uint16_t index = 0;
    uint16_t crc = 0xffff;
    uint8_t *p = _txBuf + RH_ASK_PREAMBLE_LEN; // star
    uint8_t count = len + 3 + RH_ASK_HEADER_LEN; // Ad

    if (len > RH_ASK_MAX_MESSAGE_LEN)
    return false;

    // Wait for transmitter to become available
    waitPacketSent();

    if (!waitCAD())
    return false;  // Check channel activity

    // Encode the message length
    crc = RHcrc_ccitt_update(crc, count);
    p[index++] = symbols[count >> 4];
    p[index++] = symbols[count & 0xf];

    // Encode the headers
    crc = RHcrc_ccitt_update(crc, _txHeaderTo);
    p[index++] = symbols[_txHeaderTo >> 4];
    p[index++] = symbols[_txHeaderTo & 0xf];
    crc = RHcrc_ccitt_update(crc, _txHeaderFrom);
    p[index++] = symbols[_txHeaderFrom >> 4];
    p[index++] = symbols[_txHeaderFrom & 0xf];
    crc = RHcrc_ccitt_update(crc, _txHeaderId);
    p[index++] = symbols[_txHeaderId >> 4];
    p[index++] = symbols[_txHeaderId & 0xf];
    crc = RHcrc_ccitt_update(crc, _txHeaderFlags);
    p[index++] = symbols[_txHeaderFlags >> 4];
    p[index++] = symbols[_txHeaderFlags & 0xf];

    // Encode the message into 6 bit symbols. Each byt
    // 2 6-bit symbols, high nybble first, low nybble
    for (i = 0; i < len; i++)
    {
    crc = RHcrc_ccitt_update(crc, data[i]);
    p[index++] = symbols[data[i] >> 4];
    p[index++] = symbols[data[i] & 0xf];
    }

    // Append the fcs, 16 bits before encoding (4 6-bi
    // Caution: VW expects the _ones_complement_ of th
    // VW sends FCS as low byte then hi byte
    crc = ~crc;
    p[index++] = symbols[(crc >> 4)  & 0xf];
    p[index++] = symbols[crc & 0xf];
    p[index++] = symbols[(crc >> 12) & 0xf];
    p[index++] = symbols[(crc >> 8)  & 0xf];

    // Total number of 6-bit symbols to send
    _txBufLen = index + RH_ASK_PREAMBLE_LEN;

    // Start the low level interrupt handler sending s
    setModeTx();

    return true;
}
```



```
bool RH_ASK::init()
{
    if (!RHGenericDriver::init())
    return false;
    thisASKDriver = this;

#if (RH_PLATFORM == RH_PLATFORM_GENERIC_AVR8)
 #ifdef RH_ASK_PTT_PIN
    RH_ASK_PTT_DDR  |=  (1<<RH_ASK_PTT_PIN);
    RH_ASK_TX_DDR   |=  (1<<RH_ASK_TX_PIN);
    RH_ASK_RX_DDR   &= ~(1<<RH_ASK_RX_PIN);
 #else
    RH_ASK_TX_DDR   |=  (1<<RH_ASK_TX_PIN);
    RH_ASK_RX_DDR   &= ~(1<<RH_ASK_RX_PIN);
 #endif
#else
    // Set up digital IO pins for arduino
    pinMode(_txPin, OUTPUT);
    pinMode(_rxPin, INPUT);
    pinMode(_pttPin, OUTPUT);
#endif

    // Ready to go
    setModeIdle();
    timerSetup();

    return true;
}
```

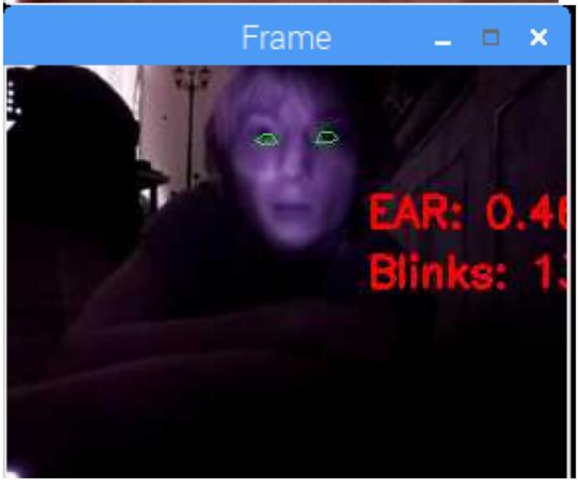
3. Light and dark facial detection



Frame

EAR: 0.3

Blinks: 2



Frame

EAR: 0.4

Blinks: 1