

Anti-theft Vehicle Tracking System

**SURJITH BHAGAVATH SINGH
MANOHAR KARTHIKEYAN**

**FINAL PROJECT REPORT
ECEN 5613 EMBEDDED SYSTEM DESIGN
DECEMBER 12, 2015**

Table of Contents

1. INTRODUCTION	3
1.1 SYSTEM OVERVIEW	3
2. TECHNICAL DESCRIPTION	3
2.1 BOARD DESIGN	4
2.1.1 Power Circuit Design	4
2.2 USB CAMERA INTERFACE	5
2.3 GPS MODULE	5
2.4 GRAPHICAL DISPLAY	6
2.5 MOTOR DRIVE CIRCUIT	7
3. FIRMWARE DESIGN.....	8
3.1 GPS DRIVERS	9
3.2 CAMERA DRIVERS	11
3.3 LCD DRIVERS	11
3.4 STEPPER MOTOR DRIVERS	11
3.5 BUZZER PWM DRIVERS	13
4. SOFTWARE DESIGN	14
4.1 FLASK WEBSERVER AND WI-FI CONFIGURATION	14
4.2 MAIN PYTHON CODE	16
4.2.1 Live feed	18
4.2.2 Track GPS Location.....	18
4.2.3 Blow the alarm	19
4.2.4 Lock the doors and Block the Engine	19
4.3 USER INTERFACE DEVELOPMENT	19
5. TESTING PROCESS	20
6. RESULTS AND ERROR ANALYSIS.....	20
7. CONCLUSION.....	21
8. FUTURE DEVELOPMENT IDEAS	21
9. ACKNOWLEDGEMENTS	22
10. REFERENCE	22
11. APPENDICES	22
11.1 APPENDIX - BILL OF MATERIALS	23
11.2 APPENDIX - SCHEMATICS	24
11.3 APPENDIX - FIRMWARE SOURCE CODE.....	27
GPS Code:.....	27
Camera Code:.....	31
11.4 APPENDIX - SOFTWARE SOURCE CODE	32
Main Code:.....	32
11.5 APPENDIX - DATA SHEETS AND APPLICATION NOTES	43

1. Introduction

This project was inspired from the latest entertainment movie fast and furious, where a set of mob with talented technicians steal costly cars for money. This is a great issue, a recent survey from FBI states that 0.7 million vehicle thefts reported nation-wide in 2013. As the students of embedded system design we decided to design and develop a system which can track the vehicle with GPS Location, audio/video feed from a camera attached to the car at a low cost (i.e. less than 100\$).

This system has a GPS module, Camera module, Audio Processing Unit (TLV320AIC3104) with a powerful ARM Sitara™ (1GHz) Processor board (Beagle Bone Black). It has an alarming system which can be activated on user's request. For easy access a web server User Interface (UI) has been developed.

1.1 System Overview

This system incorporates several hardware modules and software modules. Hardware modules are shown in the block diagram Fig1. It will give a rough idea about the system we have developed.

Beagle Bone Black being the main Processing unit which integrates other modules (GPS, Graphical display, Camera, Wi-Fi) and does the sequencing of events according to the user inputs.

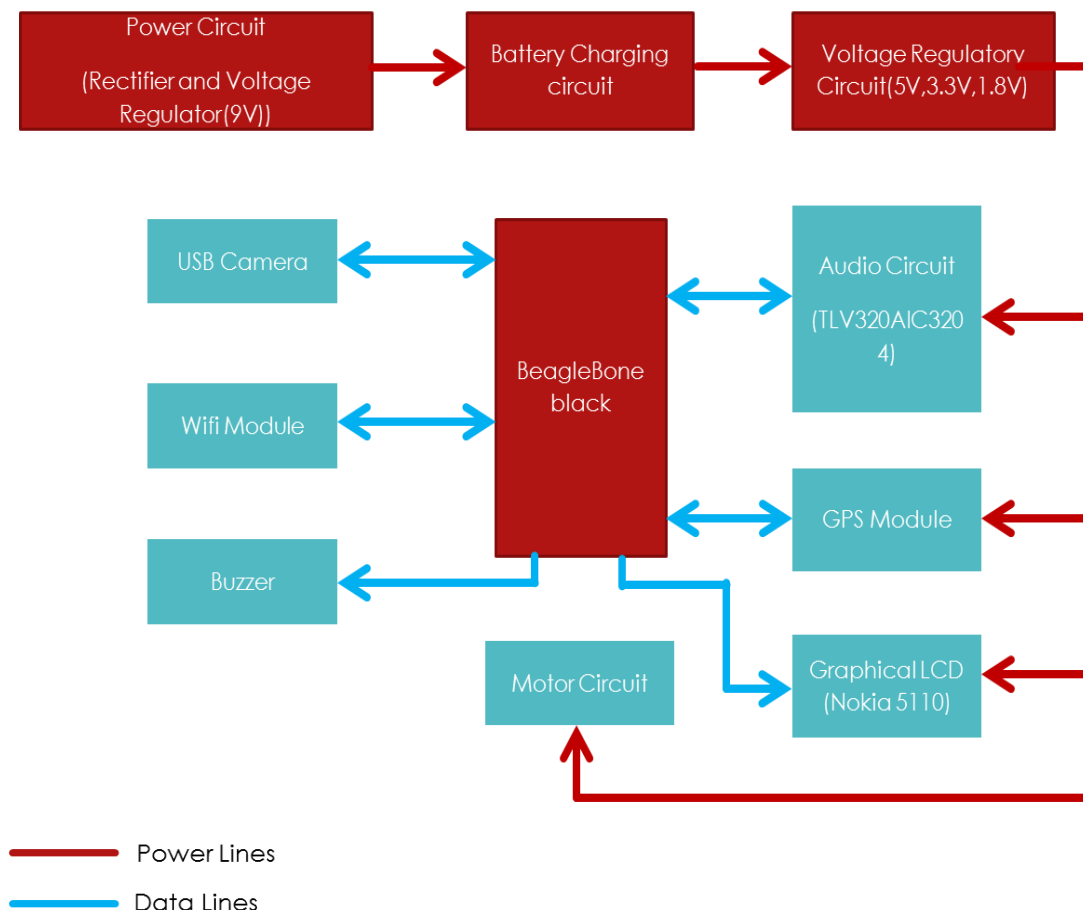


Figure 1: Block Diagram of the System

2. Technical Description

The following section details the design and implementation of the system. The sections are outlined as follows: Board design includes circuit design and new hardware components, firmware design which

depicts the driver functions implemented and the software design which details the user implementation along with Wi-Fi configuration.

2.1 Board Design

As shown in Fig 1. our system is based on ARM Sitara™ (1GHz) Processor board (Beagle Bone Black). The system has been developed in two boards. One board contains the Power and Audio circuit board and the other one is the Peripherals board which contains Beagle bone black and other hardware devices.

We designed the Power circuit by analysing the current required by each component and the battery along with the battery circuit was implemented to have uninterrupted power supply to the system.

2.1.1 Power Circuit Design

The power circuit was designed to make sure Beagle Bone black and other interfaced hardware components has the necessary voltage and current input. 12V, 3A power adapter was used along with the Bridge rectifier to get close to 11VDC without much noise. The LM7809 voltage regulator was used to bring down the voltage to constant 9V. The reason constant 9V was desired was to make sure the voltage is higher than the Battery voltage, which is 6V. Using the 9V as the input a simple battery circuit was implemented as shown in Fig 2. With a power resistor (3 ohm and 5W) and a diode to protect reverse current (i.e. to eliminate the battery to charge the power adapter in case the voltage at LM7809 drops below 9V). The charging circuit was designed to allow 1A charging current to the battery.

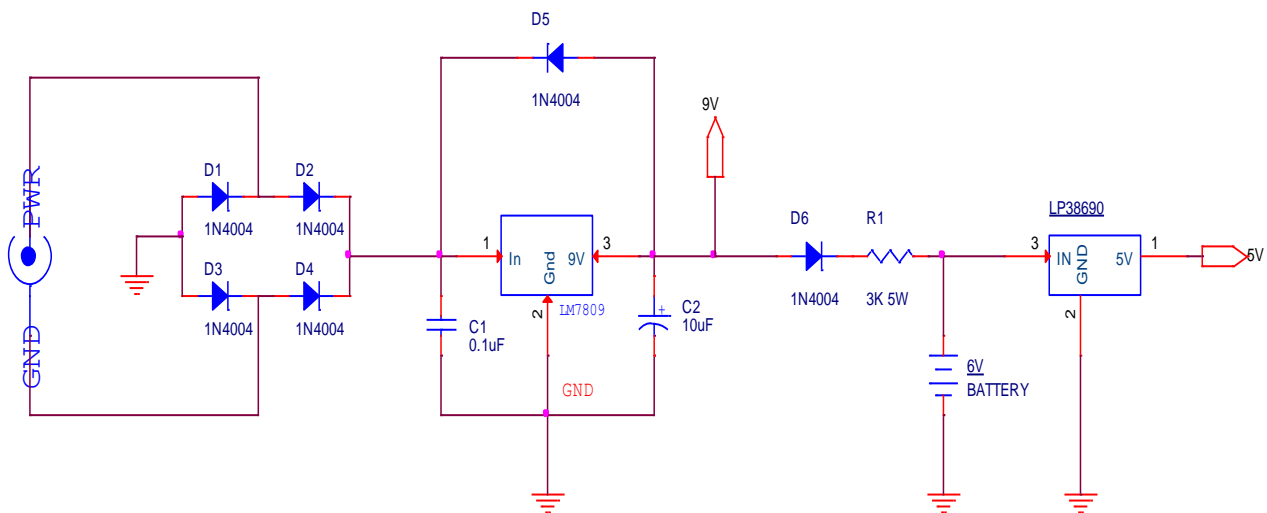


Figure 2: Battery charging circuit

To get the constant 5V which required by beagle bone black and other hardware devices, after the battery an LDO (LP 38960) was used. The reason for choosing an LDO instead of a linear voltage regulator was because the voltage of the battery tends to decrease as the battery discharges there is no room for a large dropout voltage. LP 38960 is a SMT component and so it was carefully soldered on the board using the hot airflow gun. The audio processing circuit was designed using (TLV320AIC3204) and it required 3.3V and 1.8V. For this purpose the adjustable voltage regulator (LM317) was used. The formula $V_{out} = 1.25V \left(1 + \frac{R_2}{R_1}\right) + I_{adj} * R_2$ ¹ was used from the datasheet to calculate the necessary resistor values for the 3.3V and 1.8V. The calculation was carried assuming I_{adj} was 100uA and $R_1 = 240$ ohms and input voltage as 9V. The designed circuit is shown in Fig 3. Capacitors are used as decoupling capacitors in the input and output side.

¹ <http://cdn.sparkfun.com/datasheets/Components/General/LM317TG.pdf>

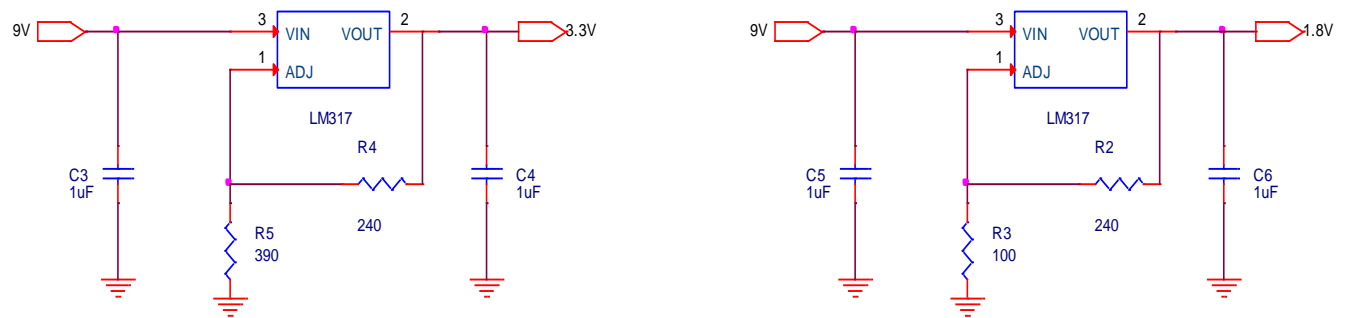


Figure 3: LM 317 circuits

2.2 USB Camera Interface

Once we decided with interfacing a camera. We were looking at I2C camera to be interfaced for the project. Since live video feed is our goal, we needed to carry out the process at 23 frames per second. The image size we were capturing was 384 x 288. The size of one frame of 384 x 288 at 72DPI and 24bit RGB bit depth was found to be 0.3MB as shown in Fig 4. Hence for 23 frames it would be 6.9MB. Since the maximum transfer rate of I2C camera was 400KB/sec², it won't be compatible for the project. So we chose USB camera which has a speed of 400MB/sec³. The USB camera is connected through Beagle Bone's USB port to capture frames.

On-Screen Photo Width:	384	pixels
On-Screen Photo Height:	288	pixels
Printed dots per inch:	72 DPI	
Bit depth:	24 bit RGB (16.7 million colors)	
<hr/>		
Total Pixels:	110592	pixels
Printed Photo Width:	5.333333333333333	inches
Printed Photo Height:	4	inches
Bytes:	331776	bytes
Kilobytes:	324	kb
Megabytes:	0.31640625	Mb

Courtesy: <http://jan.ucc.nau.edu/lrm22/pixels2bytes/calculator.htm>

Figure 4: Image size calculation

2.3 GPS Module

We used the breakout that is built around the MTK3339 chipset, high-quality GPS module that can track up to 22 satellites on 66 channels, has an excellent high-sensitivity and built in antenna. Since it requires 5V and 20mA⁴ for operation the GPS was powered from the output terminal of LP38690. The RX and TX pins of GPS module was connected to the Beagle Bone Black TX and RX respectively for

² <http://www.i2c-bus.org/speed/>

³ <http://www.pcworld.com/article/2360306/usb-3-0-speed-real-and-imagined.html>

⁴ <http://www.mouser.com/ds/2/737/adafruit-ultimate-gps-779243.pdf>

serial communication. For testing purpose we initially used to EN pin which is the enable pin in the GPS to enable and turn off when necessary. Whenever EN pin is grounded the GPS is turned off. The time received from the GPS is in UTC time zone. The GPS module was programmed to get the longitude and latitude values along with the speed.

GPS Pins	Beagle Bone Black pins
TX	P9_26 (Serial 1 RX)
RX	P9_24 (Serial 1 TX)

Table 1: Pin connection of GPS module

2.4 Graphical Display

For graphical display we used the Nokia 5110. It uses the PCD8544 controller which is designed to drive a graphic display of 48 rows and 84 columns⁵. The PCD8544 is interfaced with beagle bone black through a serial bus interface. Bit banging was performed using beagle bone black to communicate with the Graphic LCD. The Graphic LCD contains D/\bar{C} pin which enables us to select either command/address or data input and is read during the last bit of data byte. The SDIN – serial data input and SCLK – serial clock input of the Graphic LCD was connected to the GPIO pins of beagle bone black to perform bit banging. The \overline{RES} was connected to GPIO pin of beagle bone black which gave the option of externally resetting the LCD. When \overline{RES} is low it interrupts the transmission and no data is written to the RAM of LCD. The supply voltage was given from the LM317 output terminal which is 3.3V.

Data is downloaded in bytes into the 48 by 84 bits RAM data display matrix of PCD8544. The columns are addressed by the address pointer. The address ranges are: X 0 to 83 (1010011), Y 0 to 5 (101). In the vertical addressing mode ($V = 1$), the Y address increments after each. After the last Y address ($Y = 5$), Y wraps around to 0 and X increments to address the next column. In the horizontal addressing mode ($V = 0$), the X address increments after each byte. After the last X address ($X = 83$), X wraps around to 0 and Y increments to address the next row. After the very last address ($X = 83$ and $Y = 5$), the address pointers wrap around to address ($X = 0$ and $Y = 0$). Here we are using horizontal addressing mode ($V=0$) which is set during the LCD initialisation.

Signal	LCD Pins	Beagle Bone Black pins
SCE (Chip enable) (Active Low)	3	P9_17 (GPIO0_5)
RES (Reset) (Active Low)	4	P9_12 (GPIO1_28)
D/\bar{C} (Data/Command selection)	5	P9_15 (GPIO1_16)
SDIN (Serial Input)	6	P9_18 (GPIO0_4)
SCLK (Clock input)	7	P9_22 (GPIO0_2)

Table 2: Pin connection of Graphic LCD

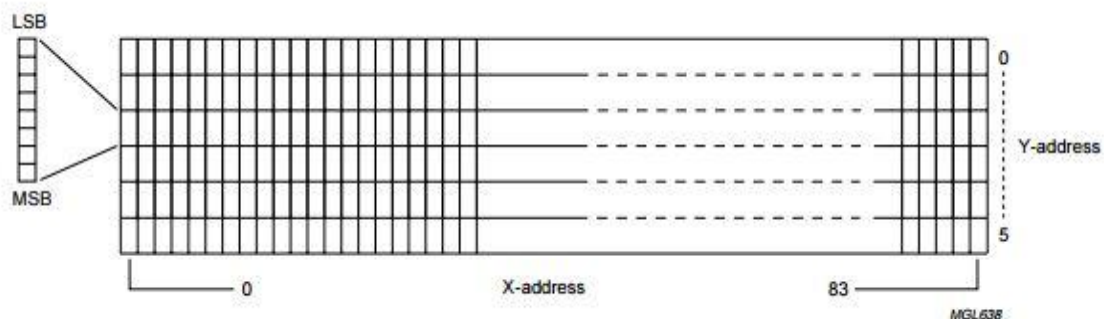
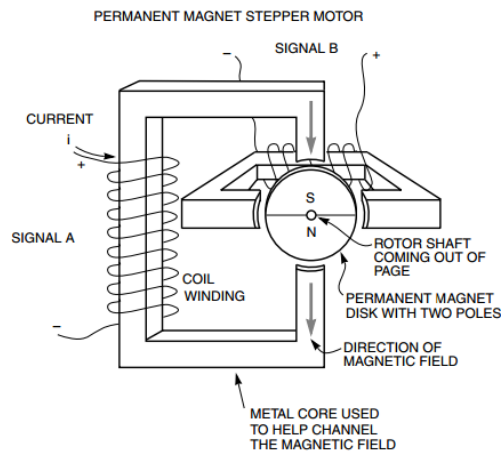


Figure 5: LCD-RAM format, addressing

⁵ <https://www.sparkfun.com/datasheets/LCD/Monochrome/Nokia5110.pdf>

2.5 Motor Driver circuit

We have mimicked the actuator mechanism which will be used to lock the door and engine, using a stepper motor in our project. The stepper motor which we are using for this project is a Permanent Magnet (PM) Stepper motor (i.e. Rotor is a permanent magnet). The magnet will align according to the polarity in the stator coils. So we have to give a sequence of pulses which can rotate it in clockwise or anticlockwise direction. The sequence is well explained in the software section. When the stepper motor is energized, it will sink up to 20mA of current and it is proportional to the load attached to the shaft. But Beagle Bone Black's maximum sink current is 8 mA for a GPIO pin.



Courtesy: http://www.nxp.com/files/microcontrollers/doc/app_note/AN2974.pdf

Figure 6: Permanent magnet stepper motor

So we have to drive the stepper motor through a motor driver circuit. ULN2003A stepper motor driver IC is used to drive the stepper motor. ULN 2003A are high voltage with seven Darlington's per package and each can output current 500mA⁶. The control pins are connected to the GPIO Pins of Beagle Bone Black. The full step operation was performed using the GPIO pins of Beagle bone black. In full step operation the motor moves through its basic step angle, i.e., a 1.8° step motor takes 200 steps per motor revolution. We have used the single phase mode⁷, also known as "one-phase on, full step" excitation in which the motor is operated with only one phase energized at a time. We used this mode since it requires the least amount of power from the driver of any of the excitation modes.

ULN2003A Pins	Beagle Bone Black pins
Pin 1 (IN 1)	P8_13 (GPIO0_23)
Pin 2 (IN 2)	P8_14 (GPIO0_26)
Pin 3 (IN 3)	P8_15 (GPIO1_15)
Pin 4 (IN 4)	P8_16 (GPIO1_14)

Table 3: Pin connection of ULN2003A

⁶ <http://www.st.com/web/en/resource/technical/document/datasheet/CD00001244.pdf>

⁷ <http://www.nmbtc.com/step-motors/engineering/full-half-and-microstepping/>

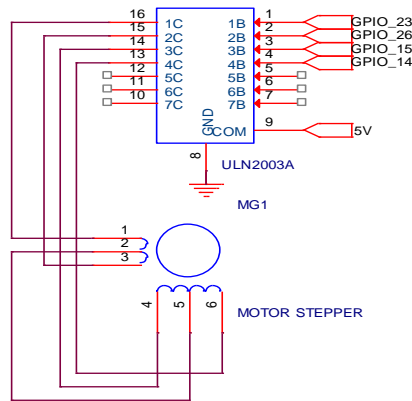


Figure 7: Schematic of motor driver circuit

2.6 Audio Circuit

The Audio processing circuit was built around TLV320AIC3104⁸ which is a low power stereo audio codec with stereo headphone amplifier, as well as multiple inputs and outputs that are programmable in single-ended or fully differential configurations.

The record path of the TLV320AIC3104 contains integrated microphone bias, digitally controlled stereo microphone preamplifier, and automatic gain control (AGC), with mix/mux capability among the multiple. During record, programmable can remove audible noise.

The serial control bus supports the I2C protocol, whereas the serial audio data bus is programmable for I2S. The schematic of the Audio circuit is shown below. We used both ceramic and electrolytic capacitors to eliminate both high frequency and low frequency noise. Unfortunately after building the circuit we blew our IC and couldn't test the circuit with Beagle Bone Black.

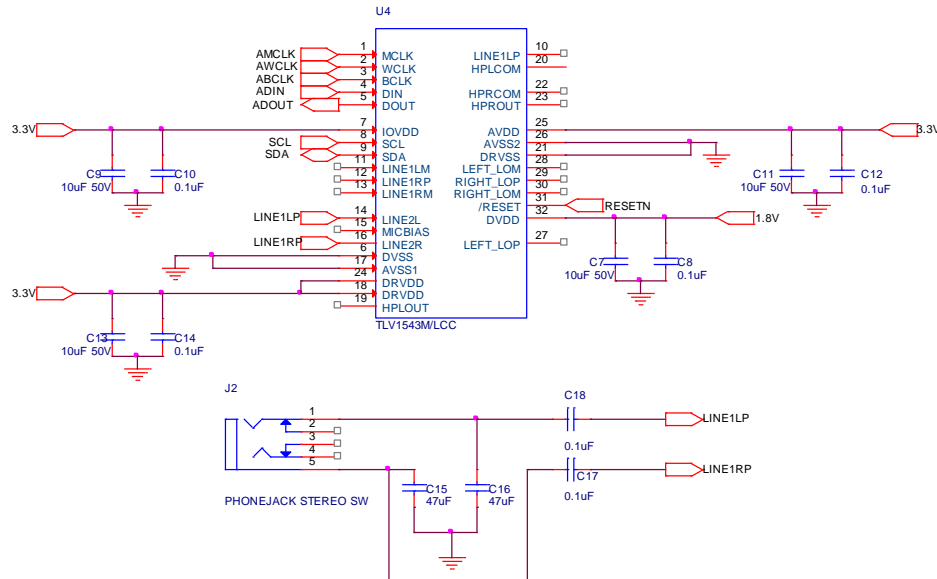


Figure 8 : Audio Circuit

3. Firmware Design

Firmware of this project includes the basic drivers for GPS (UART), Graphical display (SPI), Buzzer (PWM), Stepper motor Driver and open source USB Camera and Wi-Fi drivers.

⁸ <http://www.ti.com/lit/ds/slas510d/slas510d.pdf>

3.1 GPS Drivers

GPS module (Adafruit) has UART interface to communicate with. We are using Beagle Bone Black's UART1 for interfacing with GPS module.

We have developed a user space driver for GPS module. The code has been attached below. We have developed the code in python language. Similar to Intel 8051 microcontroller it has libraries which can be used to access basic peripherals.

GPS can be configured by sending the command characters(PMTK Commands) through UART. PMTK protocol is a set of commands that can be sent to GlobalTop MTK GPS modules to control and change its attributes⁹ and settings.

PMTK Command sets in GPS Module comprise of the following command characters. Default baud rate of the GPS module is 9600. So the configuration procedure has to be done at 9600 (Before configuring for the desired baud rate).

Preamble	Talker ID	Pkt Type	Data Field	*	CHK1	CHK2	CR	LF
----------	-----------	----------	------------	---	------	------	----	----

Table 4: PMTK GPS Command format

Field	Length	Type	Description
Preamble	1 byte	Character	“\$”
Talker ID	4 byte	Character string	“PMTK”
Pkt Type	3 byte	Character string	From “000” to “999”, an identifier used to tell the decoder how to decode the packet
Data field	Variable		A “,” must be inserted before each data field to help decoder process the Data Field
*	1 byte	Character	The star symbol is used make the end of Data Field
CHK1,CHK2	2 byte	Character String	Checksum of the data between preamble “,” and “*”
CR,LF	2 byte	Binary data	Used to identify the end of packet.(CR = ‘\r’ LF = ‘\n’)

Table 5: Description of PMTK Command fields

Purpose	PMTK Command
Baud rate settings	\$PMTK251,<BAUD RATE>*27<CR><LF>
NMEA Update rate	\$PMTK220,<UPDATE RATE in milliseconds>*1F<CR><LF>
GPS Measure rate	\$PMTK220,<UPDATE RATE in milliseconds>,0,0,0,0*1C<CR><LF>

Table 6: Description of PMTK commands used in GPS firmware code

⁹ https://www.adafruit.com/datasheets/PMTK_A11.pdf

Baud rate has to be initialized first. Measure rate and update rate has to be updated after initializing the baud rate. The gps python code's [See Appendix GPS Code:] ' __init__ ' function does this sequence. After initializing the GPS module, GPS module starts sending the NMEA sentences.

NMEA sentences are the specifications developed by National Marine Electronics Association (NMEA) for GPS receiver information. Any GPS receiver which looks for a data will expect the data to be in this format. It includes PVT (Position, Velocity and Time) information. Each sentence begins with '\$' character.

Data from GPS	Type of data
GPGGA	Fix data
GPRMC	Minimum recommended data

Table 7: Type of GPS strings Used

This is an example data string from a GPS module after initializing. This string has to be parsed according to the explanation given in Table 5. GPS python code's [See Appendix GPS Code:] 'read' function sequenced the instructions to parse the Latitude, Longitude, Speed and altitude information from the GPGGA string.

\$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47¹⁰

GPGGA String	Explanation
GGA	Global Positioning System Fix Data
123519	Fix taken at 12:35:19 UTC
4807.038,N	Latitude 48 deg 07.038' N
01131.000,E	Longitude 11 deg 31.000' E
1	Fix quality: 0 = invalid 1 = GPS fix (SPS) 2 = DGPS fix 3 = PPS fix 4 = Real Time Kinematic 5 = Float RTK 6 = estimated (dead reckoning) 7 = Manual input mode 8 = Simulation mode
08	Number of satellites being tracked
0.9	Horizontal dilution of position
545.4,M	Altitude, Meters, above mean sea level

¹⁰ <http://www.gpsinformation.org/dale/nmea.html>

46.9,M	Height of geoid (mean sea level) above WGS84 ellipsoid
(empty field)	time in seconds since last DGPS update
(empty field)	DGPS station ID number
*47	the checksum data, always begins with *

Table 8: GPGGA string explanation

UART module in the Beagle Bone Black has to be configured, by typing this command in the code (Serial ('/dev/tty01',9600)). This initializes the UART1 to 9600 baud rate.

3.2 Camera Drivers

The camera which has been used for this project is a generic UVC (USB Video Class) Camera. It makes use of the inbuilt UVC drivers. The device has been attached to the Beagle Bone through its USB port. At the device address (/dev/video0).

'fswebcam' is a webcam support package which has been developed as an open source tool for taking snaps from a web camera. It makes use of Video4Linux(V4L) libraries to take images from the camera. It stores the image at the working directory in the name 'image.png' (Refer Code below). For every one second this image will be taken and stored in the working directory with the same file name(Overwriting on the old image).

'fswebcam' is not available by default in linux images. It can be installed by using the following command in linux terminal.

```
sudo apt-get install fswebcam
```

```
sudo - Run as root (Admin Privileges)
```

```
apt-get - Package management tool in debian
```

```
install - Installs the fswebcam package into the system
```

'OpenCV(Open Source Computer Vision)' Libraries are very simple to use for image processing applications. We are using a function called 'imencode' from 'OpenCV' library, which encodes the 'image.png' to a jpeg format, then the jpeg image is converted to a string. This string will be returned from 'get_frame' function to the main function. [\[See Appendix Camera Code:\]](#)

The latest Beagle Bone Black image comes with inbuilt opencv library. So it can be imported directly to the python code.

3.3 LCD Drivers

Graphical display requires SPI interface to communicate with Beagle Bone Black, we are using General Purpose Input/Output (GPIO) Pins for SPI. We are bit banging the GPIO pin to transmit data to Graphical LCD.

The LCD Graphical display pins are connected with Beagle Bone Black as shown in Table 2. The DDRAM is a 48 x 84 bits static RAM which stores the display data. The RAM is divided into six banks of 84 bytes ($6 \times 8 \times 84$ bits). During RAM access, data is transferred to the RAM through the serial interface. The display is generated by continuously shifting rows of RAM data to the dot matrix LCD through the column outputs. Due to the temperature dependency of the liquid crystals' viscosity, the LCD controlling voltage V_{LCD} must be increased at lower temperatures to maintain optimum contrast. The temperature coefficient of V_{LCD} , can be selected from four values by setting bits TC1 and

TC0. During the initialisation the temperature coefficient is set to V_{LCD} temperature coefficient 0 by clearing the bits TC1 and TC0.

The instruction format is divided into two modes: If D/C (mode select) is set LOW, the current byte is interpreted as command byte. If D/C is set HIGH, the following bytes are stored in the display data RAM. After every data byte, the address counter is incremented automatically. The level of the D/C signal is read during the last bit of data byte. Each instruction can be sent in any order to the PCD8544. The MSB of a byte is transmitted first.

The Serial interface is initialized when \overline{SCE} is HIGH. In this state, SCLK clock pulses have no effect and no power is consumed by the serial interface. A negative edge on \overline{SCE} enables the serial interface and indicates the start of a data transmission. SDIN is sampled at the positive edge of SCLK.

A reset pulse with RES interrupts the transmission. No data is written into the RAM. The registers are cleared. If SCE is LOW after the positive edge of RES, the serial interface is ready to receive bit 7 of a command/data byte.

LCD has to be initialised at the start of the program. The initialisation is as follows¹¹

- Immediately following power-on, the contents of all internal registers and of the RAM are undefined. A \overline{RES} was applied.
- The power down control and entry mode bits is set to 0 to make the chip active and enable horizontal addressing. The Instruction set control bit is set 1 to enable extended instruction set.
- Set the temperature coefficient. In our program we have set the temperature coefficient to coefficient 0.
- We set the display mode to normal mode by setting D bit and clearing the E bit.
- The Vop was set by sending 0x88.

After the above process the LCD would be initialised and ready to display the data.

We have written a function goto_xy which would enable us to navigate to any address in the LCD. To set the X address, the given column is OR with 0x80 and to set Y address, the given row is OR with 0x40 and $D/\overline{C} = 0$.

The following steps were done to write a command or data

- \overline{RES} was set LOW and then HIGH to make sure the LCD is at known state.
- \overline{SCE} was set HIGH and then set LOW to enable the serial interface since the negative edge of the \overline{SCE} enables the serial interface.
- The D/C pin in the LCD is set or cleared accordingly with respect to data or command.
- The data is then sent to the SDIN pin in the LCD.
- The SCLK was made HIGH and then made LOW since the data is sampled in the rising edge of the SCLK.

Once all the 7 bits are received the value of D/C is then read to carry out the necessary operation.

To do LCD clear we just write 0 to all the pixels in the LCD

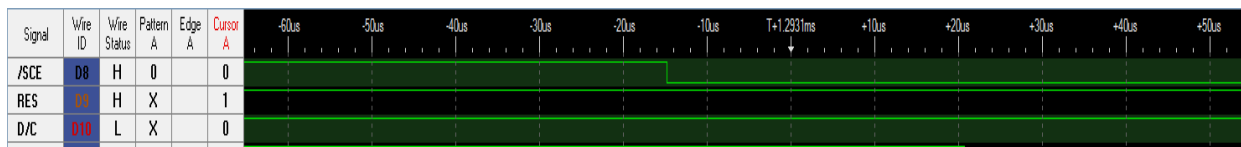


Figure 9: LCD clear

¹¹ <https://www.sparkfun.com/datasheets/LCD/Monochrome/Nokia5110.pdf>

The figure above shows waveforms captured using the logic port analyser during a LCD clear. From the waveform it can be seen that the SCE was made LOW to enable serial interface, RES was HIGH, since the operation performed is DATA, the D/C is made HIGH. The SDIN is always low since we are trying write 0 to all pixels. The SCLK is made HIGH and LOW respectively and the DATA is sampled in the positive edge of the SCLK. So DATA can change when SCLK is LOW.

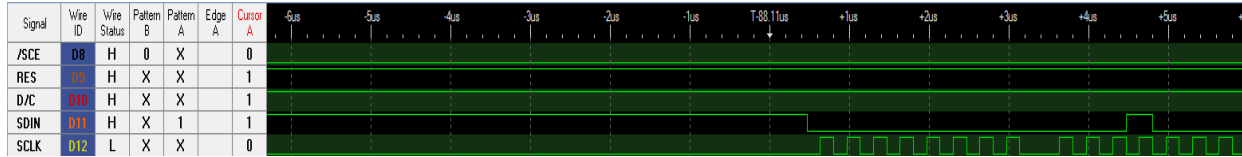


Figure 10: Writing data

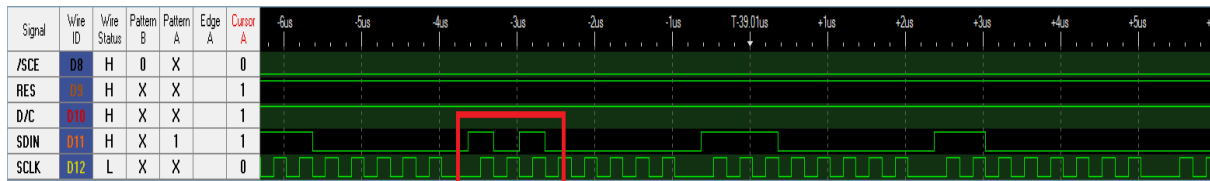


Figure 11: Writing data to display a character

The above figures 10 and 11 shows the waveforms captured when we were trying a character in the LCD. The red box in the figure 11 shows DATA is changed when SCLK is LOW, since DATA is sampled in the positive edge of the SCLK.

We are successfully perform Serial communication. [See Appendix Main Code:]

3.4 Stepper Motor Drivers

Stepper motors's control signals are a sequence of ones and zeros given in order to make it rotate. There are 4 control signals for the stepper motor. We are using single pole energizing method, so we have to one hot encode the control signal. For a clock wise rotation the sequence is 1000,0100,0010,0001. For anti-clock wise rotation the sequence is 1000,0001,0010,0100. For a complete 360 degree rotation either one of the sequence has to be done for 512 times. 360 degree is split into 512×4 elements. Resolution of this stepper motor is 0.175 degree.

Clock wise sequence has been implemented in this project. Wave form of the control signal is shown in Figure.12

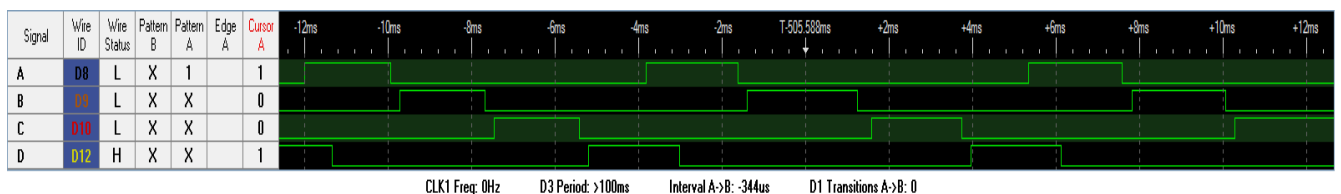
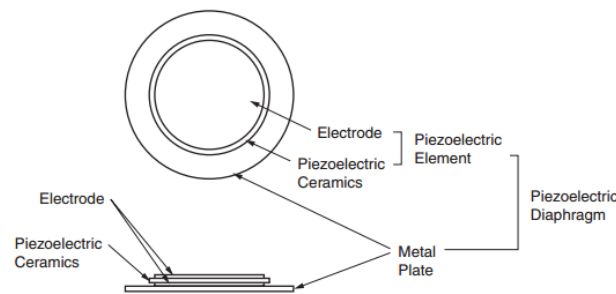


Figure 12: Stepper motor excitation from Beagle bone black

3.5 Buzzer PWM Drivers

Piezo-electric Buzzer works on DC, Piezo Diaphragm oscillates according to the voltage given across the buzzer. This voltage can be varied by giving PWM wave across it, In order to get a variable frequency.



Courtesy: <http://www.murata.com/~media/webrenewal/support/library/catalog/products/sound/p15e.ashx?la=en>

Figure 13: Piezo Buzzer Internal Structure

When a variable DC voltage (PWM) is given across the Piezo (at Electrodes), it oscillates the Piezo ceramics according to the potential difference. Which creates the sound with frequency proportional to the PWM voltage.

Pulse Width Modulated wave is given from the Beagle Bone Black's PWM Pins. Adafruit's GPIO libraries are used to give a PWM Wave. Buzzer was turned on only when the user wants to. The control is directly linked with the user Interface which is described in detail in software section.

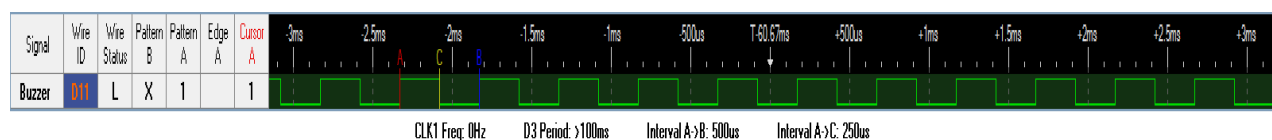


Figure 14: Waveform from P9_16

The duty cycle was set to 50%. From Figure 14, it is found that the $T_{on}=250\mu s$ and $T=500\mu s$. Hence the duty cycle was found to match 50%.

4. Software Design

4.1 Flask Webserver and Wi-Fi Configuration

Flask is a lightweight Python web framework based on Werkzeug and Jinja 2. Webserver in our project is implemented using this open source library. A web user interface, which is the basic functionality of this project. When the main python code is executed it will create a server at Beagle Bone's IP address with a port number, which can be modified in the code.

By default Beagle Bone Black has an IP address, When it is connected through USB(Ethernet through USB) is 192.168.7.2 . This cannot be modified by the user. IP address varies according to the network. If it is connected to the local network (Xfinity Wi-Fi in our house) it's IP address is 10.0.0.9. This IP address is unique and it varies according to the network. We can find this IP address by connecting it through USB and Wi-Fi being enabled. The linux terminal commands which gives the IP mentioned below.

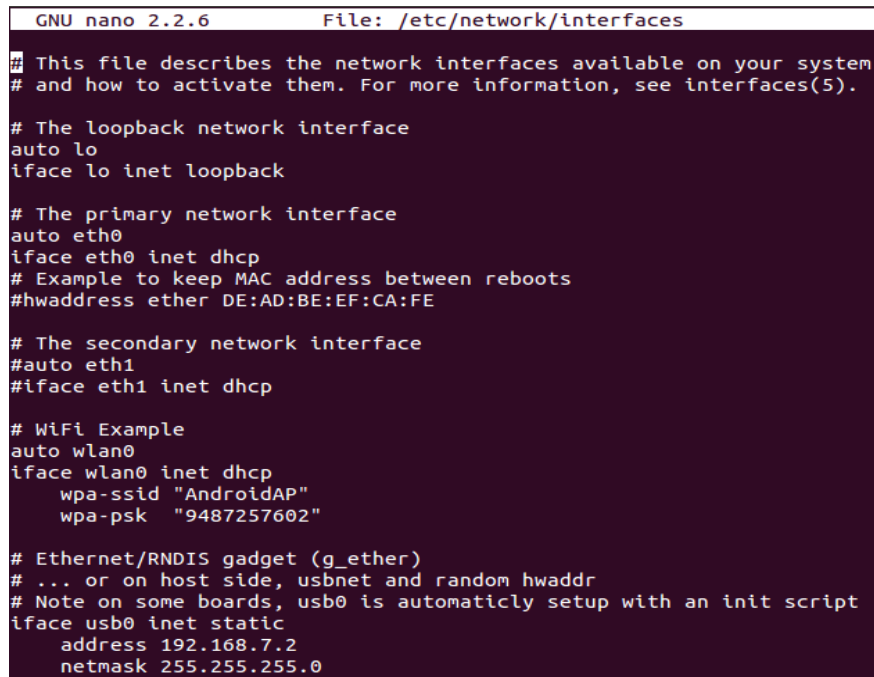
As a prototype we have used Wi-Fi to show the webserver. For independent operation a GSM/GPRS Module has to be integrated with the system.

Now Wi-Fi module attached to the Beagle Bone Black has to get connected to the predefined network.

We have to modify the network interface file in the Linux. We have an editor called nano, which is inbuilt with the Linux Debian Image. By typing the following Command it will open the interfaces file.

```
nano /etc/network/interfaces
```

In this configuration file we have to modify the SSID (Wi-Fi Network Name) and Password. After editing it, save it by pressing Ctrl+O and Ctrl+X to exit.



```

GNU nano 2.2.6      File: /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp
# Example to keep MAC address between reboots
#hwaddress ether DE:AD:BE:EF:CA:FE

# The secondary network interface
#auto eth1
#iface eth1 inet dhcp

# WiFi Example
auto wlan0
iface wlan0 inet dhcp
    wpa-ssid "AndroidAP"
    wpa-psk  "9487257602"

# Ethernet/RNDIS gadget (g_ether)
# ... or on host side, usbnet and random hwaddr
# Note on some boards, usb0 is automatically setup with an init script
iface usb0 inet static
    address 192.168.7.2
    netmask 255.255.255.0
  
```

Figure 15: Screen shot of Wi-Fi configuration



```

wlan0    Link encap:Ethernet  HWaddr 74:da:38:41:1e:29
         inet addr:192.168.43.9  Bcast:192.168.43.255  Mask:255.255.255.0
         inet6 addr: fe80::76da:38ff:fe41:1e29/64  Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:1571 errors:0 dropped:8 overruns:0 frame:0
         TX packets:5527 errors:0 dropped:1 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:209491 (204.5 KiB)  TX bytes:7808637 (7.4 MiB)

root@beaglebone:~#
  
```

Figure 16: Screen shot of ifconfig (to check IP Address)

Type the following command to check whether Wi-Fi got connected or not.

```
ifup WLAN0
```

This command makes the Wi-Fi get connected to the network if it is not connected already. To Check the IP address bounded for the device can be checked by giving the following command in its terminal.

```
ifconfig
```

This gives the IP address of all the mode of connections. Ethernet's IP address, WLAN's IP address, Subnet mask and data sent and received. We can check the IP address of the WLAN and use it to access the application.

This can be checked by typing the IP address from the other device connected to the network. We are using a port number of 5000 for our application. Once the Beagle Bone Black Boots up, the shell script which has been configured to execute the application at the start up. We have to wait for the Beagle Bone Black to Boot up. It typically takes a minute to boot up and get connected to the Wi-Fi network.

4.2 Main Python code

This is the main python code, which is calling functions and classes from other files. This code initializes the flask webserver and hosts the webpage. This code has the direct control over the entire process. After the web server is initialized, it is going to wait for the webpage request from the user.

Once the request is received it will show the webpage (192.168.43.9:5000) for the user with all other controls as shown in the Fig 14 below.

Flow chart explains the flow of the main.py code. When main.py started executing it starts the server. And waits for the request from the user. Once the request is received it executes the corresponding operation according to the user input.

Main code is attached in the appendix [\[See Appendix Main Code:\]](#). Initialization of Graphical LCD and stepper motor input output configurations are done in main code's initial stages.

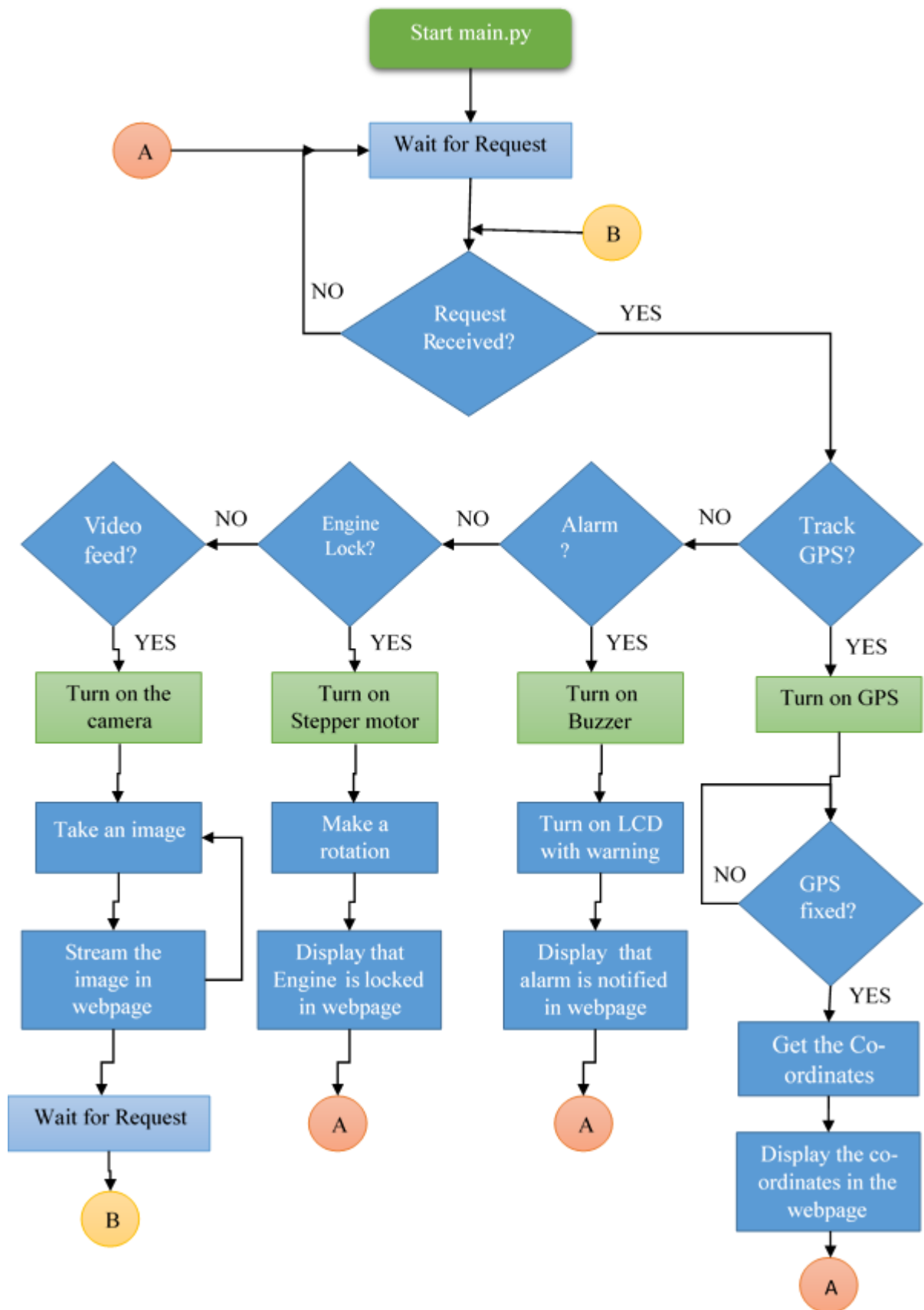


Figure 17: Flow Chart

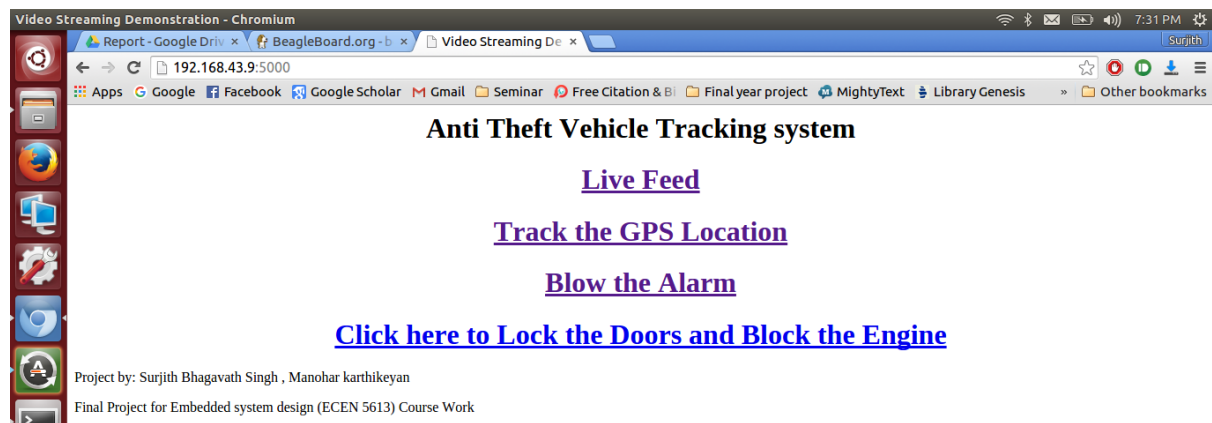


Figure 18: User interface Webpage

4.2.1 Live feed

This option opens the webpage with the address `192.168.43.9:5000/video_feed`. This page serves the image at the rate of a frame per two seconds. Python file `Camera.py` has been executed and the output of the function is being streamed at this server location. Camera can be attached near the steering facing the driver. So that it will be easy to identify the person who is driving the car when it is stolen.

In the `main.py` code, below `@app.route(/video_feed)` statement ; is the response from class `VideoCamera` is being returned to the webpage. Here is a screenshot of the output given in Fig 19.

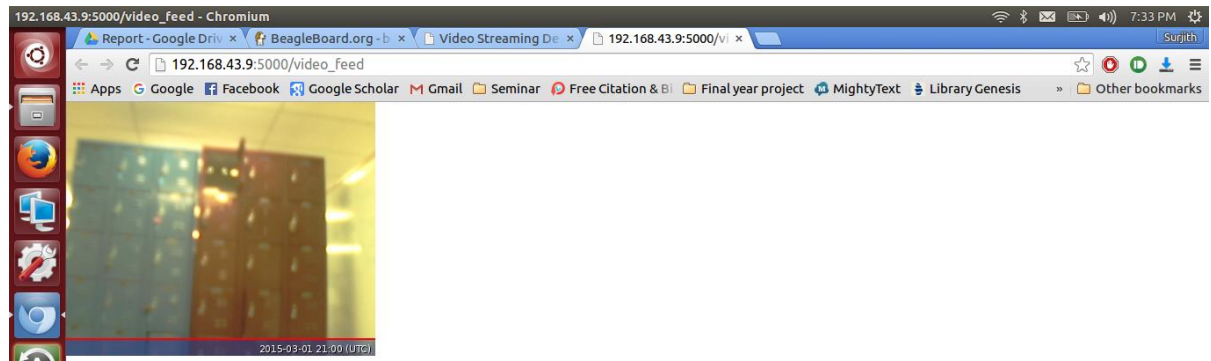


Figure 19: Video feed from the camera to the web interface

4.2.2 Track GPS Location

This link triggers the main function in `gps.py`. `gps.py` acquires the data from the GPS module and returns back the string which has all the Latitude, Longitude, speed, Altitude information. It will be shown in the web page as shown in the figure.

In the `main.py` code, below `@app.route(/locate)` statement ; is the response from main function and returns the string in HTML format to the webpage. Here is a screenshot of the output given in Fig 20.

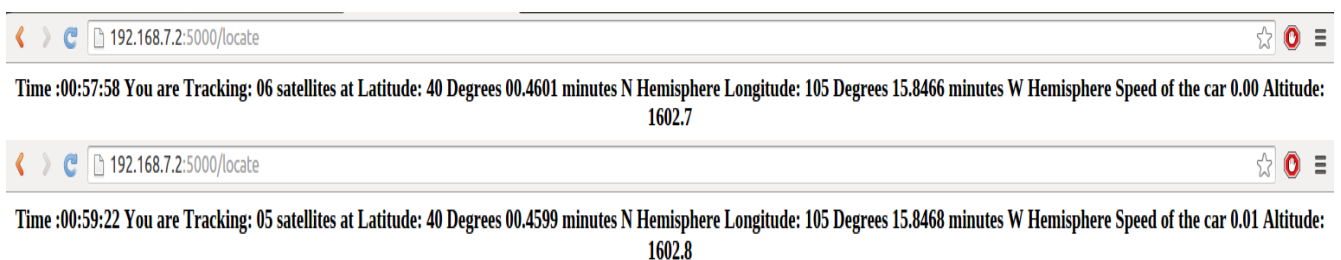


Figure 20: GPS location with UTC Time, Altitude, Speed

4.2.3 Blow the alarm

Graphical LCD and Buzzer is being attached using this interface. If the user wants to prompt the environment around the car and driver about the vehicle theft, This link can be used to trigger the alarming system attached with the vehicle. It blows the alarm and blinks the Graphical LCD with the contact details of the owner.

In the main.py code, below `@app.route(/alarm)` statement ; triggers the alarm function in main.py and returns a feedback saying that alarm has been triggered as shown in Figure 21.

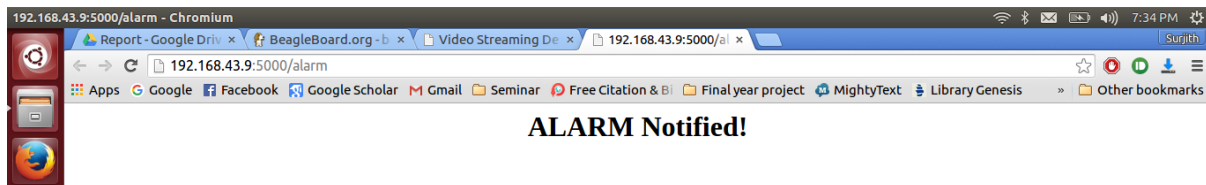


Figure 21: Alarm notification

4.2.4 Lock the doors and Block the Engine

When the user wants to lock the doors of the vehicle and block the engine from starting, it can be activated by triggering the lock function. The lock function actuates the actuator (Stepper motor connected to the Beagle Bone Black). It makes sure that the vehicle is locked and cannot be started using the regular key.

The stepper motor has been actuated by giving a sequence of pulses through it's control pins. It energizes the stepper motor coil. The sequence is shown in the main.py code. For a complete 360 degree rotation of the stepper motor that sequence has to be repeated 512 times.

In the main.py code, below `@app.route(/lock)` statement; triggers the lock function in main.py which triggers the stepper motor sequence and returns a feedback saying that doors has been locked as shown in Fig 22.



Figure 22: Mimic of actuator system (stepper motor)

4.3 User Interface Development

Since we are using the web server to serve as a user interface. We developed the user interface using HTML(Hyper Text Markup Language). We are hosting the pages according to the address entered on the web browser. This is being handled by flask web server libraries.

Each element in the user interface has a separate link linked to the separate predefined operations mentioned in the main function. HTML code is attached in the appendix.

5. Testing Process

The project was carried in a progressive way. Each module was built separately and tested before moving on to the next one. This saved lots of time. Most of the testing procedures were manual. Testing the camera module was the most difficult since the camera was a low resolution camera. Whenever we tried to capture an image with a frame size higher the camera's default size we were not able to capture any image. So by trial error method we set the frame size to be 384 x 288.

After designing the power circuit, it was built step by step. The circuit was probed at different terminals to view the waveform in the Oscilloscope and the decoupling capacitors were added to smoothen the waveforms. The currents and voltages were also measure at different operations and made sure we were not going beyond the ratings limit of any component. The current was measured during all the operations of the project were working fine and made sure the devices had enough current to perform the operation properly. The power circuit was also tested for its reliable operation by draining the battery several times and tested for its charge level.

6. Results and Error Analysis

Beagle Bone's USB port was not working when it was powered up using Battery. Later we realized the charge level in battery causes a major difference.

First we were trying the with a MicroVision USB camera which was not working properly at times, gave us a time out error. But the same code was working fine with the Logitech camera. Later we realized to change the timeout variable to the maximum.

Before testing the audio circuit's proper functionality it smoked. We later realized that we probed in between 2 pins which shorted the IC.

OpenCV libraries had version issues in the beginning, Later we uninstalled and reinstalled the package which resolved the problem.

Figuring out a way to find the IP address of the Beagle Bone Black when it is connected to local network was difficult in the beginning. Later we figured out a way to find it. IP address's assignment varies time to time. We can fix this by assigning static IP configuration in Beagle Bone Black.

We cannot be able to connect to the UCB Wireless network for a long time. Then we realized we need authentication to use these devices.

GPS module often throws some error; "GPS Not fixed" We have to take the GPS module to an open space and wait for half an hour to get it fixed and do the testing. After it gets fixed it has got quite accurate results of our location.

Initially we were developing the user interface using Qt Libraries in Linux. We developed a simple user interface for Linux environments. Later we realized in order to make it OS independent it is better to make the interface in web server.

Python language is very simple to use and it has lot of support. We wouldn't have completed our project if it we would have done it in different language.

We tried to develop the Linux kernel drivers for the audio circuit. We read through few text books and we started writing the code by taking ALSA (Advanced Linux Sound Architecture) drivers as reference. Unfortunately we don't have hardware to check this driver. We spent our Thanksgiving week on Linux Drivers. It was a good learning Experience.

Interfacing the camera with Beagle Bone Black took us some time, we had to configure lot of things inside Linux, and we had to install few packages in order to make it work. Power issue was a big issue.

Sometimes camera module won't work, we do wonder about it because at similar conditions it had worked before. Later when we connected a power source which has constant current output (2 Ampere rated adapter) or fully charged battery and it worked like a charm.

Beagle Bone Black was rebooting often when everything is working fine. Beagle Bone Black has a PMIC which was the cause of the problem. We connected Graphical LCD, GPS module and the USB Hub (Wi-Fi and Camera) with the Beagle Bone Black (i.e. Taking power directly from Beagle Bone Black's Power Pins). We solved this issue later by sourcing the external peripherals Graphical LCD and GPS module through the Power circuit which we have developed. USB Peripherals were still working on Beagle Bone's Power Source.

Bit Banging the SPI through GPIO pins was kind of Hard, Initially we used open source libraries to check our modules, Later we developed our own functions. It could have been easier in C language. Nokia 5110 Graphical LCD is similar to what we used in Embedded System Design Coursework. With few Extra Features. Backlight LED Brightness was controlled using PWM.

7. Conclusion

We have reached the conclusion part finally, It was a very good experience exploring the other aspects of embedded systems, We have learnt a new operating system Linux, which is a whole new world to learn about. Firmware development for a new platform at the beginning was challenging, embedded system design course work had taught us about the nooks and corners on how to study an embedded platform from scratch. We have learnt a lot of application notes which we have attached in the reference section.

We were happy while working on this project as well as the outcome is overwhelming. We feel this is a good project which quenched out thirst towards Embedded systems. We got a chance to develop firmware for Graphical LCD, Stepper motors GPS module. We worked on the software for camera and Wi-Fi Modules. It was challenging working on power circuits.

8. Future Development Ideas

Instead of Wi-Fi; GSM/3G/4G Module can be integrated so that it can be accessed anywhere in the world.

Audio Circuit should have been implemented using VS1053, VS1053 has lot of support in terms of Application notes, Tutorials and detailed documentation is available for project development.

Single PCB should have been designed and integrated all the modules together.

Camera has to be changed to Logitech HD Camera, it has lot of support and documentation for image processing projects.

We have mimicked the implementation of Engine and Door Locking system, Future development should be analyse the actuators used in real car and implement.

We have developed a simple prototype, to demonstrate the concept and to explore the concepts of Embedded System. It can be developed as a product with Industry Standards.

PMIC for the entire system can be included in the system. Mechanical aspect of the design has to be considered in order to make it smaller, invisible among the other components in the automobile

9. Acknowledgements

We would like to thank Professor. Linden McClure for giving us this wonderful opportunity to explore on this project and for helping us shape the project.

We wish to thank ITLL for providing lending us the camera, and workspace to work on our project.

We would like to thank various authors of application notes, tutorials and other various sources cited below. We specially want to thank Sparkfun and Adafruit for their well-documented tutorials for GPS module and Graphical LCD.

10. Reference

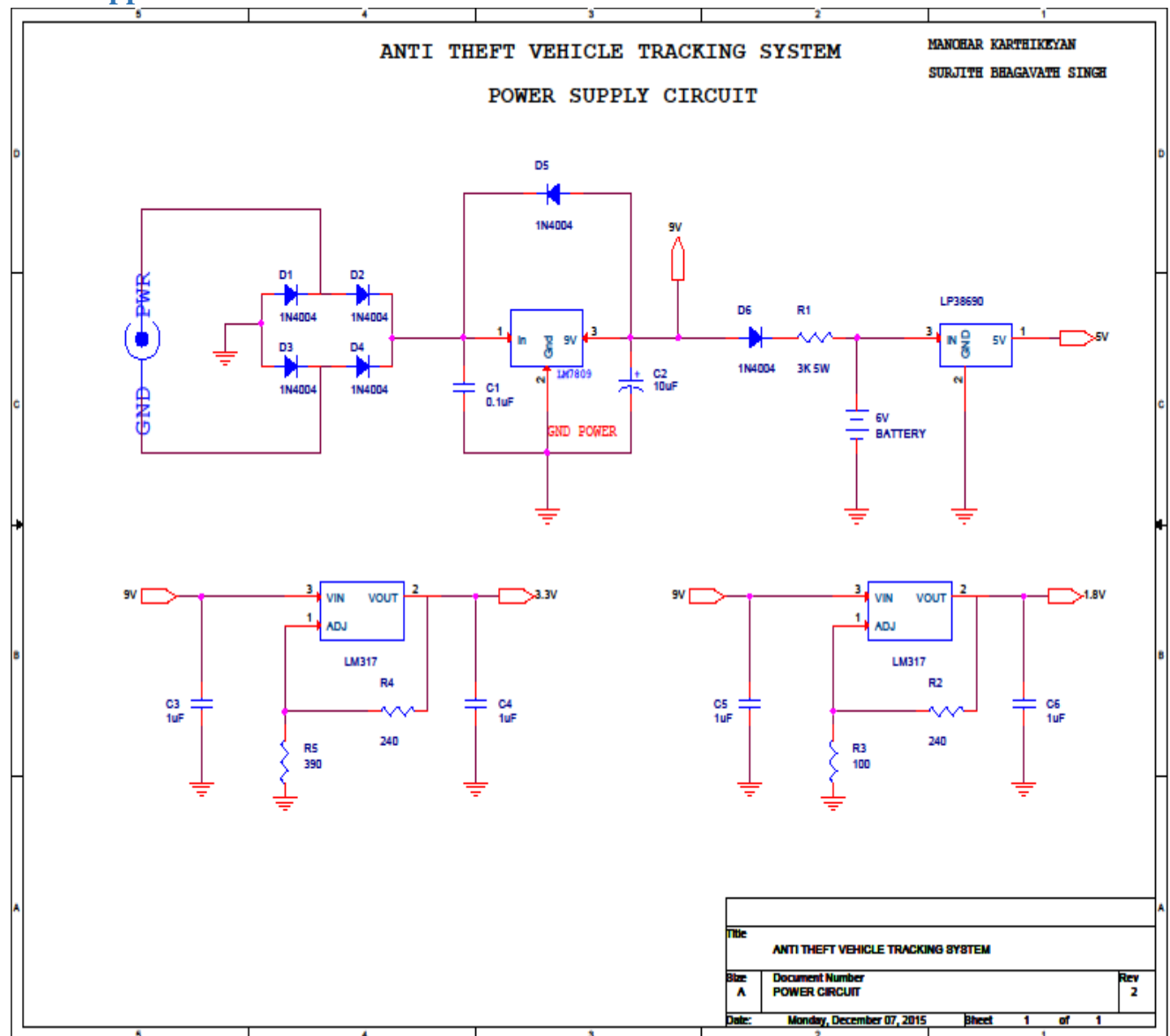
- [1] Cdn.sparkfun.com, 'Sparkfun Nokia 5110 LCD', 2015. [Online]. Available: http://cdn.sparkfun.com/datasheets/LCD/Monochrome/Nokia_5110_Example.pde. [Accessed: 13-Dec- 2015].
- [2] A. Rosebrock, 'Install OpenCV 3 and Python 2.7+ on Ubuntu - PyimageSearch', PyImageSearch, 2015. [Online]. Available: <http://www.pyimagesearch.com/2015/06/22/install-opencv-3-0-and-python-2-7-on-ubuntu/>. [Accessed: 13- Dec- 2015].
- [3] Adafruit, 'Nokia 5110 LCD Python Library', 2015. [Online]. Available: <https://learn.adafruit.com/downloads/pdf/nokia-5110-3310-lcd-python-library.pdf>. [Accessed: 13-Dec- 2015].
- [4] P. Salzman, The Linux Kernel Module Programming Guide, 1st ed. 2015.
- [5] M. Grinberg, 'Video Streaming with Flask - miguelgrinberg.com', Blog.miguelgrinberg.com, 2014. [Online]. Available: <http://blog.miguelgrinberg.com/post/video-streaming-with-flask>. [Accessed: 13-Dec- 2015].
- [6] Learn.adafruit.com, 'WiFi | BeagleBone | Adafruit Learning System', 2015. [Online]. Available: <https://learn.adafruit.com/beaglebone/wifi>. [Accessed: 13- Dec- 2015].
- [7] Tuptechboy.com, 'Beaglebone Black | Technology Tutorials', 2015. [Online]. Available: <http://www.tuptechboy.com/beaglebone-black/>. [Accessed: 13- Dec- 2015].
- [8] K. Kwong, 'Audio Project On Beagle Bone Black', kmingk, 2013. [Online]. Available: <http://kmingk.com/audio-project-on-beagle-bone-black/>. [Accessed: 13- Dec- 2015].
- [9] Texas Instruments, 'Design and Configuration Guide for the TLV320AIC3204 and TLV320AIC3254 Audio Codecs . Application Note-SLAA404C', 2015. [Online]. Available: <http://www.ti.com/lit/an/sl404c/sl404c.pdf>. [Accessed: 13- Dec- 2015].
- [10] J. Corbet, A. Rubini and G. Kroah-Hartman, Linux Device Drivers, 3rd ed. Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman, 2015.
- [11] Kernel Drivers - Free Electrons Tutorial, 1st ed. 2015.
- [12] Details on speed of i2c [Online]
Available: <http://www.i2c-bus.org/speed/>
- [13] Details on speed of USB [Online] Available: <http://www.pcworld.com/article/2360306/usb-3-0-speed-real-and-imagined.html>
- [14] <http://www.gpsinformation.org/dale/nmea.html>
- [15] Guide on getting started with Beagle bone [Online] Available: <http://beagleboard.org/>

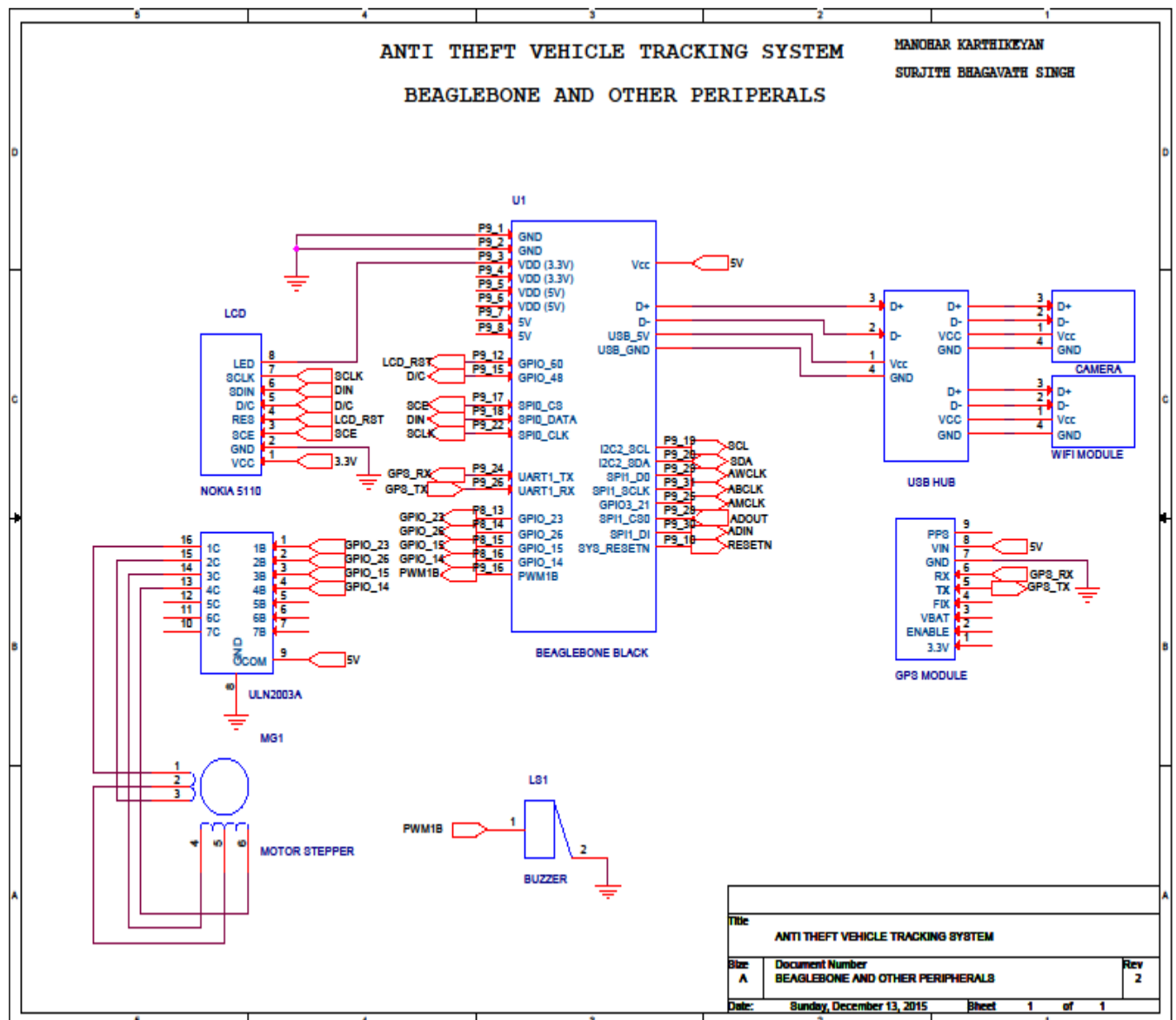
11. APPENDICES

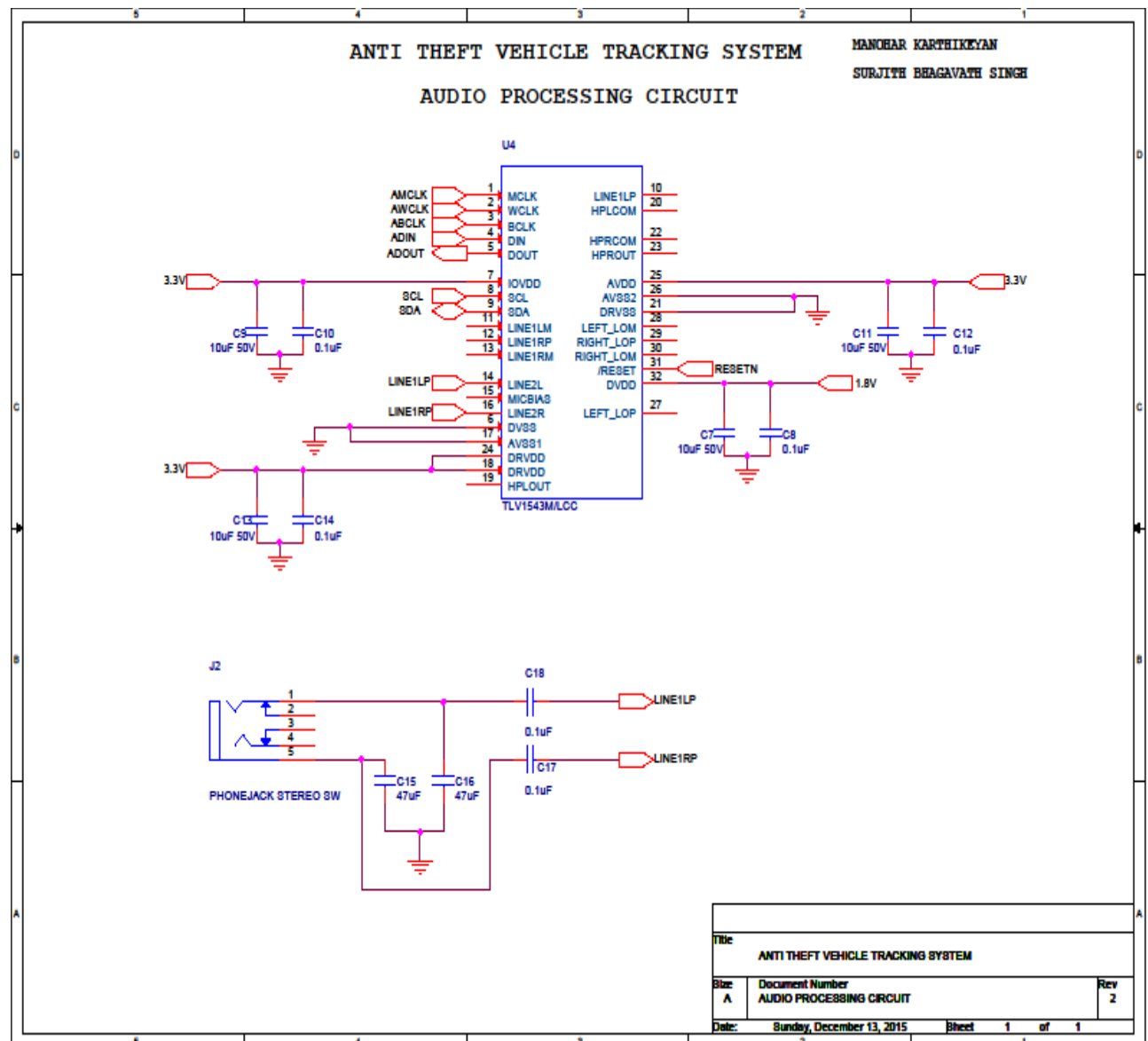
11.1 Appendix - Bill of Materials

Part Description	Source	Cost
1N4004 (6)	Digi-Key www.digikey.com	\$0.90
Power Jack	Digi-Key www.digikey.com	\$1.00
LM 7809	Digi-Key www.digikey.com	\$0.56
LM 317 (2)	Digi-Key www.digikey.com	\$1.06
LP38690	Texas Instruments – Ordered as Sample www.ti.com	\$0.00
TLV320AIC3104	Texas Instruments – Ordered as Sample www.ti.com	\$0.00
Batteries and Battery Pack	Courtesy : ITLL	\$0.00
USB Camera	Courtesy : ITLL	\$0.00
Buzzer	Courtesy : ITLL	\$0.00
Graphic LCD 84x48 – Nokia 5110	Sparkfun www.sparkfun.com	\$9.95
Audio Jack	JB Saunders, Boulder	\$1.00
240 ohm Resistors (2)	JB Saunders, Boulder	\$0.15
3 ohm 5 W Resistor	JB Saunders, Boulder	\$0.50
100 ohm and 390 ohm Resistors	JB Saunders, Boulder	\$0.30
QFN-32 to DIP-32 SMT Adapter	Proto Advantage www.proto-advantage.com	\$8.39
Perfboards (2)	E-Store, CU Boulder	\$8.00
Jumper Wires	Sparkfun www.sparkfun.com	\$4.00
Adafruit Ultimate GPS Breakout	Amazon www.Amazon.com	\$39.00
0.1uF(3) and 10uF ceramic capacitors	JB Saunders, Boulder	\$0.60
1uF ceramic capacitors (4)	JB Saunders, Boulder	\$0.90
47uF ceramic capacitors (2)	JB Saunders, Boulder	\$0.30
10uF, 50V Electrolytic capacitors (4)	JB Saunders, Boulder	\$2.00
Mic	JB Saunders, Boulder	\$1.00
ULN2003A Motor Driver	Digi-Key www.digikey.com	\$0.50
Stepper Motor	Sparkfun www.sparkfun.com	\$6.00
Edimax WiFi Adapter	Sparkfun www.sparkfun.com	\$10.00
USB Hub	Amazon www.Amazon.com	\$6.50
TOTAL		\$102.71

11.2 Appendix - Schematics







11.3 Appendix - Firmware Source Code

GPS Code:

```
#####
# File name: gps.py
# Purpose   : Initialize the GPS module, Functionality to configure #
#             GPS module and extract the information obtained from #
#             GPS module to human readable form.
# Return    : Returns a string with latitude, longitude co=ordinates#
# Author    : Surjith Bhagavath Singh, Manohar Karthikeyan
#           : Open source Adafruit library for UART has been used #
#####

#importing the libraries for Beagle Bone Black
import serial
import Adafruit_BBIO.UART as UART
from time import sleep
UART.setup("UART1")

#Configuring UART1 for 9600 Baudrate
ser=serial.Serial('/dev/ttyO1',9600)

#####
# Class name: GPS
# Purpose    : Contains initialization function, Read function
#
#####

class GPS:

#####
# Function name: __init__
# Purpose       : Initializing the parameters to configure the GPS #
#                 module(Baudrate, rate of measurement, rate of #
#                 report using PMTK statements
#
#####

    def __init__(self):
        #Initialization

        #This set is used to set the rate the GPS reports
        UPDATE_1_sec= "$PMTK220,1000*1F\r\n"

        #setting the GPS to take measurements @ 1 sec rate
        MEAS_1_sec = "$PMTK300,1000,0,0,0,0*1C\r\n"

        #Set the Baud Rate of GPS as 57600
        BAUD_57600 = "$PMTK251,57600*2C\r\n"

        #Commands for which NMEA Sentences are sent
        ser.write(BAUD_57600)
        sleep(1)
        ser.baudrate=57600
```

```

#Send GPRMC AND GPGGA Sentences
ser.write(UPDATE_1_sec)

GPRMC_GPGGA="$PMTK314,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0*28\r\n"
sleep(1)
ser.write(MEAS_200_msec)
sleep(1)
ser.write(GPRMC_GPGGA)
sleep(1)
ser.flushInput()
ser.flushInput()
print "GPS Initialized"

#####
# Function name: read
# Purpose      : Reads the data from UART and parse the information#
#               from GPRMC and GPGGA sentences and stores it in  #
#               separate array variables                          #
#               #####
def read(self):
    ser.flushInput()
    ser.flushInput()

    #Reading NMEA sentences from GPS module
    while ser.inWaiting()==0:
        pass
    self.NMEA1=ser.readline()
    while ser.inWaiting()==0:
        pass
    self.NMEA2=ser.readline()

    #Splitting the NMEA sentence
    NMEA1_array=self.NMEA1.split(',')
    NMEA2_array=self.NMEA2.split(',')

    #Parsing the data Longitude, Latitude, Hemisphere, Altitude

    if NMEA1_array[0]=='$GPRMC':
        self.timeUTC=NMEA1_array[1][:8]+'-'+NMEA1_array[1][-6:-4]
        self.latDeg=NMEA1_array[3][:7]
        self.latMin=NMEA1_array[3][-7:]
        self.latHem=NMEA1_array[4]
        self.lonDeg=NMEA1_array[5][:7]
        self.lonMin=NMEA1_array[5][-7:]
        self.lonHem=NMEA1_array[6]
        self.knots=NMEA1_array[7]

    if NMEA1_array[0]=='$GPGGA':
        self.fix=NMEA1_array[6]
        self.altitude=NMEA1_array[9]

```

```

        self.sats=NMEA1_array[7]

        if NMEA2_array[0]=='$GPRMC':
            self.timeUTC=NMEA2_array[1][:8]+'':'+NMEA1_array[1][-8:-6]+'':'+NMEA1_array[1][-6:-4]
            self.latDeg=NMEA2_array[3][:7]
            self.latMin=NMEA2_array[3][-7:]
            self.latHem=NMEA2_array[4]
            self.lonDeg=NMEA2_array[5][:7]
            self.lonMin=NMEA2_array[5][-7:]
            self.lonHem=NMEA2_array[6]
            self.knots=NMEA2_array[7]

        if NMEA2_array[0]=='$GPGLL':
            self.fix=NMEA2_array[6]
            self.altitude=NMEA2_array[9]
            self.sats=NMEA2_array[7]

#####
# Function name: main
# Purpose      : Calls the initialization function, read function
#               and parse the information into human readable
#               string format.
#####

def main():

    #assigning class
    myGPS=GPS()

    #infinite loop
    while(1):

        #Calling GPS read function
        myGPS.read()

        #Printing the NMEA sentences for Debugging purposes
        print myGPS.NMEA1
        print myGPS.NMEA2

        #Checks for GPS Fix
        if myGPS.fix!=0:
            print 'Universal Time: ',myGPS.timeUTC
            print 'You are Tracking: ',myGPS.sats,'
satellites'

            print 'My Latitude: ',myGPS.latDeg, 'Degrees
', myGPS.latMin,' minutes ', myGPS.latHem
            print 'My Longitude: ',myGPS.lonDeg, 'Degrees
', myGPS.lonMin,' minutes ', myGPS.lonHem
            print 'My Speed: ', myGPS.knots
            print 'My Altitude: ',myGPS.altitude

```

```
#Compiling a string to display in the web UI

    string = "Time :"+ str(myGPS.timeUTC) + " You
are Tracking: "+ str(myGPS.sats) +" satelllites"+" at Latitude:
"+str(myGPS.latDeg)+ " Degrees "+str(myGPS.latMin)+" minutes "+
str(myGPS.latHem)+" Hemisphere"+" Longitude: "+str(myGPS.lonDeg)+ "
Degrees "+str(myGPS.lonMin)+" minutes "+ str(myGPS.lonHem)+"
Hemisphere"+" Speed of the car "+str(myGPS.knots)+" Altitude:
"+str(myGPS.altitude)

#Returning the string
return string
```

Camera Code:

```
#####
# File name: camera.py
# Purpose : Initialize the USB camera and takes the image using
#           fswebcam utility in the Linux Debian distribution.
#           It Encodes the image into string using openCV python
#           libraries .
# Return : Returns a string equivalent of the image
# Author : Surjith Bhagavath Singh, Manohar Karthikeyan
#           OpenCV library has been used
#####

#Importing the openCV and system libraries
import cv2
import os

#####
# Class name: VideoCamera
# Purpose : Contains get_frame function
#
#####

class VideoCamera(object):

#####
# Function name: get_frame
# Purpose : Initializing the USB camera and taking a picture
#           using fswebcam and storing it as "image.png"
# Return : Returns the string format of the image for
#           streaming purpose.
#
#####

    def get_frame(self):
        cmd='fswebcam image.png'
        #Entering the command from linux terminal
        os.system(cmd)
        #Reading the image from the local directory
        image=cv2.imread('image.png')
        #encoding the image to jpeg format
        ret, jpeg = cv2.imencode('.jpg', image)
        #Converting the image in jpeg format to string
        return jpeg.tostring()
```


11.4 Appendix - Software Source Code

Main Code:

```
#####
# File name: main.py
# Purpose : This is the main python code, which is calling
#           functions and classes from other files. This code
#           initializes the #flask webserver and hosts the webpage#
# Return : Returns a string with latitude, longitude co=ordinates#
# Author : Surjith Bhagavath Singh, Manohar Karthikeyan
#           Open source Adafruit libraries for GPIO and PWM has
#           been used
#####

from flask import Flask, render_template, Response
from camera import VideoCamera
from gps import main
import Adafruit_BBIO.PWM as PWM
import time
import Adafruit_BBIO.GPIO as GPIO
import os

#Graphic LCD Pin declaration
RST = "P8_3"
D_C = "P8_5"
CS = "P8_7"
DIN = "P8_9"
CLK = "P8_11"

#Graphic LCD variables
CONTROL_DATA = 1          #Data Selection
CONTROL_CMD = 0           #command Selection
LCD_X = 84                #length of LCD
LCD_Y = 48                #width of LCD

#This table contains the hex values that represent pixels
#for a font that is 5 pixels wide and 8 pixels high
ASCII = ([[0x00, 0x00, 0x00, 0x00, 0x00] # 20
```

```

, [0x00, 0x00, 0x5f, 0x00, 0x00] # 21 !
, [0x00, 0x07, 0x00, 0x07, 0x00] # 22 "
, [0x14, 0x7f, 0x14, 0x7f, 0x14] # 23 #
, [0x24, 0x2a, 0x7f, 0x2a, 0x12] # 24 $
, [0x23, 0x13, 0x08, 0x64, 0x62] # 25 %
, [0x36, 0x49, 0x55, 0x22, 0x50] # 26 &
, [0x00, 0x05, 0x03, 0x00, 0x00] # 27 '
, [0x00, 0x1c, 0x22, 0x41, 0x00] # 28 (
, [0x00, 0x41, 0x22, 0x1c, 0x00] # 29 )
, [0x14, 0x08, 0x3e, 0x08, 0x14] # 2a *
, [0x08, 0x08, 0x3e, 0x08, 0x08] # 2b +
, [0x00, 0x50, 0x30, 0x00, 0x00] # 2c ,
, [0x08, 0x08, 0x08, 0x08, 0x08] # 2d -
, [0x00, 0x60, 0x60, 0x00, 0x00] # 2e .
, [0x20, 0x10, 0x08, 0x04, 0x02] # 2f /
, [0x3e, 0x51, 0x49, 0x45, 0x3e] # 30 0
, [0x00, 0x42, 0x7f, 0x40, 0x00] # 31 1
, [0x42, 0x61, 0x51, 0x49, 0x46] # 32 2
, [0x21, 0x41, 0x45, 0x4b, 0x31] # 33 3
, [0x18, 0x14, 0x12, 0x7f, 0x10] # 34 4
, [0x27, 0x45, 0x45, 0x45, 0x39] # 35 5
, [0x3c, 0x4a, 0x49, 0x49, 0x30] # 36 6
, [0x01, 0x71, 0x09, 0x05, 0x03] # 37 7
, [0x36, 0x49, 0x49, 0x49, 0x36] # 38 8
, [0x06, 0x49, 0x49, 0x29, 0x1e] # 39 9
, [0x00, 0x36, 0x36, 0x00, 0x00] # 3a :
, [0x00, 0x56, 0x36, 0x00, 0x00] # 3b ;
, [0x08, 0x14, 0x22, 0x41, 0x00] # 3c <
, [0x14, 0x14, 0x14, 0x14, 0x14] # 3d =
, [0x00, 0x41, 0x22, 0x14, 0x08] # 3e >
, [0x02, 0x01, 0x51, 0x09, 0x06] # 3f ?
, [0x32, 0x49, 0x79, 0x41, 0x3e] # 40 @
, [0x7e, 0x11, 0x11, 0x11, 0x7e] # 41 A
, [0x7f, 0x49, 0x49, 0x49, 0x36] # 42 B

```

```

, [0x3e, 0x41, 0x41, 0x41, 0x22] # 43 C
, [0x7f, 0x41, 0x41, 0x22, 0x1c] # 44 D
, [0x7f, 0x49, 0x49, 0x49, 0x41] # 45 E
, [0x7f, 0x09, 0x09, 0x09, 0x01] # 46 F
, [0x3e, 0x41, 0x49, 0x49, 0x7a] # 47 G
, [0x7f, 0x08, 0x08, 0x08, 0x7f] # 48 H
, [0x00, 0x41, 0x7f, 0x41, 0x00] # 49 I
, [0x20, 0x40, 0x41, 0x3f, 0x01] # 4a J
, [0x7f, 0x08, 0x14, 0x22, 0x41] # 4b K
, [0x7f, 0x40, 0x40, 0x40, 0x40] # 4c L
, [0x7f, 0x02, 0x0c, 0x02, 0x7f] # 4d M
, [0x7f, 0x04, 0x08, 0x10, 0x7f] # 4e N
, [0x3e, 0x41, 0x41, 0x41, 0x3e] # 4f O
, [0x7f, 0x09, 0x09, 0x09, 0x06] # 50 P
, [0x3e, 0x41, 0x51, 0x21, 0x5e] # 51 Q
, [0x7f, 0x09, 0x19, 0x29, 0x46] # 52 R
, [0x46, 0x49, 0x49, 0x49, 0x31] # 53 S
, [0x01, 0x01, 0x7f, 0x01, 0x01] # 54 T
, [0x3f, 0x40, 0x40, 0x40, 0x3f] # 55 U
, [0x1f, 0x20, 0x40, 0x20, 0x1f] # 56 V
, [0x3f, 0x40, 0x38, 0x40, 0x3f] # 57 W
, [0x63, 0x14, 0x08, 0x14, 0x63] # 58 X
, [0x07, 0x08, 0x70, 0x08, 0x07] # 59 Y
, [0x61, 0x51, 0x49, 0x45, 0x43] # 5a Z
, [0x00, 0x7f, 0x41, 0x41, 0x00] # 5b [
, [0x02, 0x04, 0x08, 0x10, 0x20] # 5c \
, [0x00, 0x41, 0x41, 0x7f, 0x00] # 5d ]
, [0x04, 0x02, 0x01, 0x02, 0x04] # 5e ^
, [0x40, 0x40, 0x40, 0x40, 0x40] # 5f _
, [0x00, 0x01, 0x02, 0x04, 0x00] # 60 `
, [0x20, 0x54, 0x54, 0x54, 0x78] # 61 a
, [0x7f, 0x48, 0x44, 0x44, 0x38] # 62 b
, [0x38, 0x44, 0x44, 0x44, 0x20] # 63 c
, [0x38, 0x44, 0x44, 0x48, 0x7f] # 64 d

```

```
, [0x38, 0x54, 0x54, 0x54, 0x18] # 65 e
, [0x08, 0x7e, 0x09, 0x01, 0x02] # 66 f
, [0x0c, 0x52, 0x52, 0x52, 0x3e] # 67 g
, [0x7f, 0x08, 0x04, 0x04, 0x78] # 68 h
, [0x00, 0x44, 0x7d, 0x40, 0x00] # 69 i
, [0x20, 0x40, 0x44, 0x3d, 0x00] # 6a j
, [0x7f, 0x10, 0x28, 0x44, 0x00] # 6b k
, [0x00, 0x41, 0x7f, 0x40, 0x00] # 6c l
, [0x7c, 0x04, 0x18, 0x04, 0x78] # 6d m
, [0x7c, 0x08, 0x04, 0x04, 0x78] # 6e n
, [0x38, 0x44, 0x44, 0x44, 0x38] # 6f o
, [0x7c, 0x14, 0x14, 0x14, 0x08] # 70 p
, [0x08, 0x14, 0x14, 0x18, 0x7c] # 71 q
, [0x7c, 0x08, 0x04, 0x04, 0x08] # 72 r
, [0x48, 0x54, 0x54, 0x54, 0x20] # 73 s
, [0x04, 0x3f, 0x44, 0x40, 0x20] # 74 t
, [0x3c, 0x40, 0x40, 0x20, 0x7c] # 75 u
, [0x1c, 0x20, 0x40, 0x20, 0x1c] # 76 v
, [0x3c, 0x40, 0x30, 0x40, 0x3c] # 77 w
, [0x44, 0x28, 0x10, 0x28, 0x44] # 78 x
, [0x0c, 0x50, 0x50, 0x50, 0x3c] # 79 y
, [0x44, 0x64, 0x54, 0x4c, 0x44] # 7a z
, [0x00, 0x08, 0x36, 0x41, 0x00] # 7b [
, [0x00, 0x00, 0x7f, 0x00, 0x00] # 7c |
, [0x00, 0x41, 0x36, 0x08, 0x00] # 7d ]
, [0x10, 0x08, 0x08, 0x10, 0x08] # 7e ~
, [0x78, 0x46, 0x41, 0x46, 0x78]]])
```

#Graptic LCD Pin cofiguration

```
GPIO.setup(RST,GPIO.OUT)
GPIO.setup(D_C,GPIO.OUT)
GPIO.setup(CS,GPIO.OUT)
GPIO.setup(DIN,GPIO.OUT)
GPIO.setup(CLK,GPIO.OUT)
```

```
#####
#Function name: lcd_init
#Purpose      : Initializing the parameters to configure the
#               Graphic LCD (Using extended command list, setting
#               entry mode, powerdown control temperature
#               coefficient, bias system, setting the contrast and
#               display mode)
#####

def lcd_init():
    print "init"
    GPIO.output(RST, GPIO.LOW)
    GPIO.output(RST, GPIO.LOW)
    spi_write(0x21, CONTROL_CMD)    #Use Extended commands
    spi_write(0xB0, CONTROL_CMD)    #Setting the contrast
    spi_write(0x04, CONTROL_CMD)    #Temparature coefficient
    spi_write(0x14, CONTROL_CMD)    #Bias system
    spi_write(0x20, CONTROL_CMD)    #Display mode
    spi_write(0x0C, CONTROL_CMD)    #Normal Display mode

#####
#Function name: spi_write
#Purpose      : The function writes data to display RAM or command
#               instruction depending on the CONTROL variable
#####

def spi_write(DATA,CONTROL):
    #The Graphic LCD is selected by giving a low signal to CS
    GPIO.output(CS, GPIO.LOW)
    #Check if the CONTROL variable is Data or Command
    if CONTROL:
        GPIO.output(D_C, GPIO.HIGH)
    else:
        GPIO.output(D_C, GPIO.LOW)
    temp_data = DATA
    #for loop for writing 8 bits
    for x in range (1,9):
        #wait for 0.01seconds
        time.sleep(0.01)
        #computation to send one bit at a time starting from MSB
```

```

    send_bit = temp_data & 0x080 #data is AND with 0x80
    if send_bit == 0x80:
        GPIO.output(DIN, GPIO.HIGH)
    elif send_bit == 0x00:
        GPIO.output(DIN, GPIO.LOW)

    # Clock is made HIGH,data sampled at positive clock edge.
    GPIO.output(CLK, GPIO.HIGH)
    time.sleep(0.01)

    #Clock is made low
    GPIO.output(CLK, GPIO.LOW)

    #the data is left shifted
    temp_data = DATA << x

    # LCD is disabled.
    GPIO.output(CS, GPIO.HIGH)

#####
#Function name: goto_xy
#Purpose      : The function enables to go to the required location#
#              in the LCD
#####
def goto_xy(x,y):
    #loading column #for setting the X address, DB7 is set to 1
    spi_write(0x80|x,CONTROL_CMD)

    #loading row for setting Y address,DB7=1 DB6,DB5,DB4 = 0
    spi_write(0x40|y,CONTROL_CMD)

#####
#Function name: lcd_clear
#Purpose      : clears the LCD display by wroing 0 in all addresses#
#####
def lcd_clear():
    #go to the first address of the LCD
    goto_xy(0,0)

    #writes zeros in all the address
    #the address gets incremented automatically if data is written
    for z in range (0,(LCD_X*LCD_Y/8)):
        spi_write(0x00,CONTROL_DATA)

```

```

        #go back to start of address

        goto_xy(0,0)

#####
#Function name: lcd_char                                     #
#Purpose       : writes a character to display by calling the #
#               spi_write function                          #
#####
def lcd_char(char_data):
    #the data is until the 5 pixels wide character is done
    for p in range (0,5):
        #Manipulating ASCII character in ASCII Array
        spi_write(ASCII[ord(char_data)-0x20][p],CONTROL_DATA)

#####
#Function name: lcd_string                                   #
#Purpose       : writes a string to display by calling the  #
#               lcd_char function                            #
#####
def lcd_string(char_data):
    i=0;
    #lcd_char function is called until the end of the string
    while (char_data[i]!='\0')
        lcd_char(char_data[0])
        i=i+1

#Stepper motor Pin configuration
GPIO.setup("P8_13", GPIO.OUT)
GPIO.setup("P8_14", GPIO.OUT)
GPIO.setup("P8_15", GPIO.OUT)
GPIO.setup("P8_16", GPIO.OUT)

#initializing the LCD
lcd_init()
lcd_clear()
input_str = "Program has started"
#go the address 1,1 in the lcd

```

```

goto_xy(1,1)
#display the string in the LCD
lcd_string(input_str)

#Initializing the flask server
app = Flask(__name__)
#Defining default page '/'
@app.route('/')

#####
#Function name: index
#Purpose      : renders the index page
#####

def index():
    #rendering index page
    return render_template('index.html')

#####
#Function name: alarm
#Purpose      : Blows the Buzzer and LCD with warning string
#####

def alarm():
    #Enabling PWM for Buzzer and LCD_Backlight
    PWM.start("P9_14", 50, 1000, 1)
    #Clearing the LCD
    lcd_clear()
    #Setting the cursor to 0,0
    goto_xy(0,0)
    input_str = "Theft Detected!"
    #display the string in the LCD
    lcd_string(input_str)
    goto_xy(0,1)
    input_str = "Contact Surjith"
    #display the string in the LCD
    lcd_string(input_str)
    goto_xy(0,2)

```



```

input_str = "@ 720-238-3307"

#display the string in the LCD
lcd_string(input_str)


#Blinking the backlight and turning on and off the buzzer
for x in range(0, 3):
    PWM.stop("P9_14")
    PWM.start("P9_16", 50,2000,0)
    time.sleep(0.5)
    PWM.start("P9_14",50,2000,0)
    PWM.stop("P9_16")
PWM.stop("P9_16")


#####
#Function name: lock
#Purpose      : rotates the stepper motor by giving the sequence
#####

def lock():
    # 512 sequence of 1000,0100,0010,0001 has to be given for 360
    # degree rotation

    for x in range(0,512):
        GPIO.output("P8_13", GPIO.HIGH)
        GPIO.output("P8_14", GPIO.LOW)
        GPIO.output("P8_15", GPIO.LOW)
        GPIO.output("P8_16", GPIO.LOW)
        time.sleep(0.001)
        GPIO.output("P8_13", GPIO.LOW)
        GPIO.output("P8_14", GPIO.HIGH)
        GPIO.output("P8_15", GPIO.LOW)
        GPIO.output("P8_16", GPIO.LOW)
        time.sleep(0.001)
        GPIO.output("P8_13", GPIO.LOW)
        GPIO.output("P8_14", GPIO.LOW)
        GPIO.output("P8_15", GPIO.HIGH)
        GPIO.output("P8_16", GPIO.LOW)

```

```

time.sleep(0.001)

GPIO.output("P8_13", GPIO.LOW)
GPIO.output("P8_14", GPIO.LOW)
GPIO.output("P8_15", GPIO.LOW)
GPIO.output("P8_16", GPIO.HIGH)
time.sleep(0.001)

#####
#Function name: gen(camera)                                     #
#Purpose       : gets the data from camera.py and stores the data in#
#               frame                                             #
#####

def gen(camera):
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n' b'Content-Type: image/jpeg\r\n\r\n'
+ frame + b'\r\n\r\n')

#defining route for locate GPS
@app.route('/locate')

#####
#Function name: ggps
#Purpose       : gets the data from gps.py and stores the data in  #
#               string and returns the string to webpage           #
#####

def ggps():
    string = main()
    return '<h3 align="center">' + string + '</h3>'

#defining route for alarm
@app.route('/alarm')

#####
#Function name: notify alarm
#Purpose       : executes alarm function and notifies the user     #
#####

def notify_alarm():
    alarm()
    return '<h1 align="center"> ALARM Notified!</h1>'

```

```

#defining route for engine lock

@app.route('/lock')

#####
#Function name: lock_stepper                                     #
#Purpose       : executes lock function and notifies the user   #
#####

def lock_stepper():
    lock()

    return '<h1 align="center"> Engine and Doors have been locked
</h1>'

#defining route for video feed

@app.route('/video_feed')

#####
#Function name: video_feed                                       #
#Purpose       : gets the reponse from camera and streams the data #
#               in webpage                                       #
#####

def video_feed():
    return Response(gen(VideoCamera()),
                    mimetype='multipart/x-mixed-replace; boundary=frame')

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)

```

11.5 Appendix - Data Sheets and Application Notes

[1] LM317 datasheet [Online].

Available: <http://cdn.sparkfun.com/datasheets/Components/General/LM317TG.pdf>

[2] The ultimate GPS breakout datasheet[Online]

Available: <http://www.mouser.com/ds/2/737/adafruit-ultimate-gps-779243.pdf>

[3]<http://www.st.com/web/en/resource/technical/document/datasheet/CD00001244.pdf>

[4] https://www.adafruit.com/datasheets/PMTK_A11.pdf

[5] Texas Instruments, 'Design and Configuration Guide for the TLV320AIC3204 and TLV320AIC3254 Audio Codecs . Application Note-SLAA404C', 2015. [Online].

Available: <http://www.ti.com/lit/an/slaa404c/slaa404c.pdf>. [Accessed: 13- Dec- 2015].

[6]Nokia 5110 datasheet [Online]

Available: <https://www.sparkfun.com/datasheets/LCD/Monochrome/Nokia5110.pdf>