5.0 Image/firmware

5.1 Image Creation Process for Raspbian OS on 32 GB SCHD Card

The SCHD card that was used for MAIDS did not come with the Raspbian operating system installed on it. In order to install the Raspbian OS image onto the card, the NOOBS and balenaEtcher software are used. To begin, the NOOBS zip archive is downloaded from https://www.raspberrypi.org/downloads/noobs/. Once NOOBS has been downloaded, the SCHD card must be formatted. The formatting of the card is done with balenaEtcher which is a free and open-source utility used for writing image files onto storage media to create live SD cards and USB flash drives. Next, the files from the NOOBS zip archive are extracted onto a directory. Finally, all the files are selected and dragged onto the SCHD card. Once the files have been copied over, the card can be ejected and used on the MAIDS device.

5.2 Firmware General Description and Requirements

The MAIDS projects incorporates its own custom-made firmware to run and control the functionality of the alarm system. Firmware refers to the firmware program, composed of individual instructions that are programmed onto the hardware device. In MAIDS case, the firmware provides the needed instructions for the computer hardware to communicate and control the PCB board components; it is stored in the 32 GB SCHD card of the device.

The development of the firmware takes into account the need to meet the real-world requirements of the project:

1. Provide a visual alarm through the dual LED module.
2. Provide an audible alert through audible warning and intrusion messages.
3. Provide rapid-response multi-channel alarm notifications (Email w/ photo, push notification, SMS messages and phone calls) in real-time.
4. Provide sound detection for gunfire, loud voices and/or breaking of glass.
5. Provide a remote method for testing visual alarm component functionality
6. Provide Internet-based database (MySQL) information for authorities use for prosecution.

Moreover, the firmware is implemented via version 3.6 of the Pyhton programming language. Python is a multi-purpose, high-level, interpreted programming language. The MAIDS project's firmware requires the following minimum requirements to run: an Intel Atom® processor or higher, at least one gigabyte of disk space, two gigabytes of RAM (recommended), the Raspbian (Buster) operating system (kernel version 4.19) and a Linux- 64-bit x86 system architecture.

The hardware's inter-component interaction (Raspberry Pi and PCB board sensors) is mediated via the RPI.GPIO library. The GPIO (general-purpose input/output) pins on a Raspberry Pi (with a 40-pin GPIO header) interfaces with the physical devices of the PCB board (i.e. button, LED module, motion sensor, sound sensor, etc.). The RPi.GPIO library handles the interface with these pins and allows the user, programmatically, to implement their function and control the device at run time.

5.3 Firmware and Hardware Interaction Description

The following block diagram demonstrates the general hardware and firmware interaction of the MAIDS project; its operative functionality is fully detailed below.

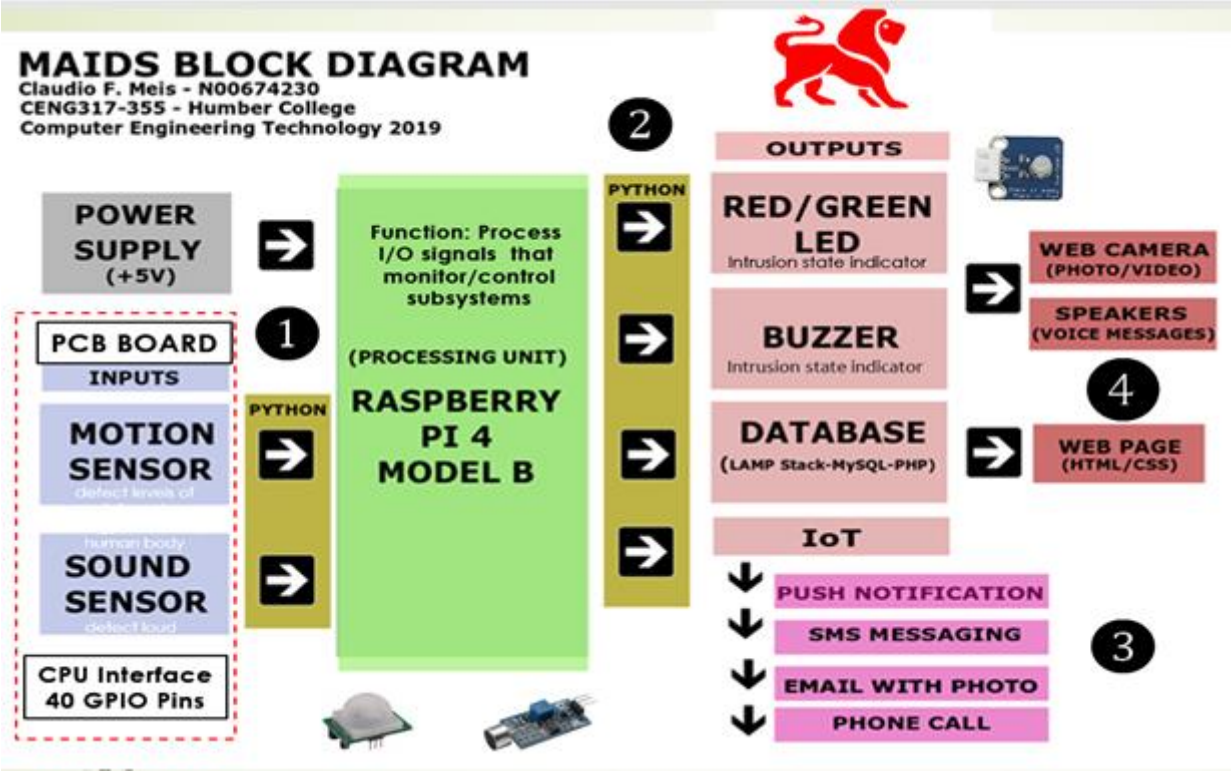

FIGURE 1 BLOCK DIAGRAM OF HARDWARE AND FIRMWARE FOR MAIDS.

In collaboration, MAIDS hardware and firmware work as follows:

- Start of section ❶ of firmware. Pressing the rocking switch to the ON position powers and activates the MAIDS system, and generates a signal to the green-LED output GPIO pin.
- Then, the firmware runs the welcoming screen and pertinent MAIDS system information function.

- Thereafter, the program displays the message: “MAIDS Surveillance Mode - Arming...” Once the system is armed, it warns the user by means of an audible message to clear the room and a countdown commences (10 down to 1).
- Upon conclusion of the countdown, the program displays the message: “MAIDS Surveillance Mode - Armed and Active...” At this point, MAIDS is activated and actively surveilling the space around it.
- Start of section ❷ of firmware. If motion or sound is detected by the motion (PIR) or sound sensors on the MAIDS PCB board, they will generate an input signal. The motion/sound input signal travel through the connecting wires to the GPIO pins (setup with the BOARD numbering system) to the Raspberry Pi 4 Model B and the firmware processes the input signal on the particular input GPIO pin and provides the following response outputs: (Start of section ❸ of firmware)

    ii. An audible message plays stating that either a motion or sound intrusion has been detected, that it is unlawful and that authorities are being notified.

    iii. The green LED then turns OFF and the red LED light is turned ON, after which it will flash intermittently ON/OFF five times.

    iv. Then, the message is displayed: “Sending push notification to Android phone...” After the message, via the Internet, the Pushover server is reached and originates the push notification to the user’s Android phone.

    v. Afterwards, another message is displayed: “Sending SMS notification to Android phone...”; again via the Internet, a Twilio (cloud communications platform as a service company) SMS server is reached and a SMS message is generated and sent to the user’s Android phone along with a

confirmation message that is displayed on the screen (i.e. “Message sent ID: SM79c3534ddeb648ee898b7d4a0e76872f”)

vi. The firmware then displays the message “Sending Intrusion Alert to email...” and initiates the following sequence of events:

- Logs into the user’s email server with the user’s username and password.
- Runs a program named ‘fswebcam’ to take a picture of the intruder and adds intrusion information (time, place, address and alert message) onto the picture’s caption.
- Then, generates the email with picture which is then sent, along with an intrusion alert message, to the user’s email.
- At this point, the message “Calling Android phone...” is displayed and a phone called is placed to the user via Twilio server; a confirmation of the call made is displayed on the screen (i.e. “Phone call ID: CA3c525429de7a5145939b559285c96095”

vii. Start of section ❹ of firmware. Immediately after the email is sent, the message “Appending Intrusion Alert to Database...” is displayed and the following takes place:

- A record of the intrusion stating intrusion id, time, place, person to contact, contact’s phone number and email is generated and saved onto the following databases:
  - MySQL Text-based database
  - MySQL GUI-based database (entries accessible via browser)

- Once the entry is recorded in the database, the program displays the following message: “Record inserted successfully into MAIDS-DB; MySQL connection – CLOSED”

viii. Finally, a siren audible is blasted through the connected speakers to frighten the intruder.

5.4 Firmware's Functional Design

Programmatically, MAIDS firmware is composed of fourteen Python functions. A function is a block of code that only runs when it is called. Two of the functions (motion and sound) require input parameters, the rest, do not. The firmware includes:

1. The set_up_gpio() function which sets up the GPIO numbering system, input and output pins and disables warnings.
2. The maids() function which displays a welcome screen and firmware information (i.e. license, author, version, etc.).
3. The notify() function which updates the database, sends email and sends push notification to android phone.
4. The alarm() function which sets the led flashing pattern.
5. The alert() function which flashes a visual alarm, plays an audible intrusion message and sounds the alarm audible.
6. The sound(sound) function which sets up the sound sensor's functionality.
7. The motion(motion) function which sets up the motion sensor's functionality.
8. The send_mail() function which sends an email alert with picture to recipient.
9. The send_androidpush() function which sets up the android push notification information with the pushover service.
10. The appendtodb() function which appends the intrusion data to a text database.
11. The intruderwarning() function which plays an intrusion warning message.
12. The siren() function which plays a siren alarm sound.
13. The activationwarning() function which sets up an audible activation countdown.

14. The dbinsert() function which inserts the intrusion information into the MySQL database.

Furthermore, the functionality of the firmware code is derived through the use of the following Python modules:

1. RPi.GPIO
2. time and datetime
3. smtplib
4. ssl
5. os
6. http.client
7. urllib
8. vlc
9. mysql.connector
10. email (with specific module imports email.mime.multipart, email.mime.base, email.mime.text and email.utils)
11. subprocess

No detailed description of each module's functionality is presented due to their lengthy nature. However, quick online searches can easily provide this information, if needed.

Finally, database creation and implementation required the use of interpolated MySQL programming commands (i.e. SELECT, INSERT, CREATE TABLE, etc.).

5.5 Code runs via CLI or remote desktop

Remote desktop is a firmware feature which permits users to connect to a remote computer, meaning a computer that is situated in a different geographic location, and interact with it as if the computer is located in front of the user. People use remote desktop access for an assortment of tasks, including: access a workplace computer from home or while roaming, access a home computer from any location, fix computer problems, perform administrative tasks or demonstrate a process or a software application.

In regards to MAIDS, one can connect to the device either locally or remotely, be it from another part of the same room or building, or from halfway around the world. As long as the MAIDS device is connected to the same network or to the Internet, the code for the MAIDS project can be run either through the Raspberry Pi command line interface (CLI) or through a remote desktop. In MAIDS case, remote desktop access refers to the secure access of the MAIDS device through the Android application running on an Android phone, tablet or through the execution of the Python source code from the CLI of another remote computer.

5.6 Wireless connectivity

Wireless connectivity is achieved via configuration of the wpa_supplicant.conf file. This file is created manually as ‘root’ and saved in the /etc/wpa_supplicant directory. The wpa_supplicant file is configured using a simple text file that specifies all accepted networks and security policies, including pre-shared keys. The file manages the configuration and administration of wireless networks on Raspbian Linux OS so that the MAIDS device can connect to configured and available wireless network. The wpa_supplicant “… implements WPA key negotiation with a WPA Authenticator and EAP authentication with Authentication Server. In addition, it controls the roaming and IEEE 802.11 authentication/association of the wireless LAN driver.” {Malinen, 2020, wpa_supplicant(8) - Linux man page}It runs as a background daemon that controls the wireless connection. The wpa_supplicant “…automatically selects the best network based on the order of network blocks in the configuration file, network security level (WPA/WPA2 is preferred), and signal strength.” {SysTutorials, 2019, wpa_supplicant.conf (5) - Linux Man Pages}The MAIDS project makes use of Wi-Fi Protected Access (WPA, also known as the IEEE 802.11i-2004 standard.) which is a type of encryption (protocols and security certifications) employed to secure the mainstream of Wi-Fi networks; a unique encryptions key (Pre-Shared Key that is 64 hexadecimal digits long) is used in WPA for every wireless client accessing the network. The main purpose for the use of encryption is to protect the confidentiality of data and to assist in the protection of its integrity. In addition, it is worth noting that WPA2 negligibly impacts network performance because of the extra processing load of encryption and decryption during every connection.

The network configuration (of the wpa_supplicant.conf file) for the MAIDS project is presented fully in Appendix II of this document and found on the GitHub repository with link:

1. https://github.com/srgawain2264/CENG317-MAIDS_PROJECT/blob/master/CENG355/wpa_supplicant.conf

In the MAIDS case, actual wireless connectivity is accomplished through the use of the RealVNC software. Real Virtual Network Computing (RealVNC) is a desktop sharing system that allows for the remote control of another computer; it can be downloaded from https://www.realvnc.com/en/connect/download/vnc. RealVNC works by transmitting all keyboard and mouse movements from one computer to another. All that is required to run RealVNC is a network TCP/IP connection, a VNC server (Installed on the MAIDS device), and a RealVNC viewer (installed on any other device used to communicate with the MAIDS device, i.e. Android tablet or phone) in order to connect to the computer that is running the server. The RealVNC Server application must be running on the device that is being controlled (MAIDS) before a connection is attempted. Once the RealVNC Server is running on the computer to be controlled, it will appear as a selectable option in RealVNC Viewer. A login screen will appear on the device connecting to the server and then the login information (for the MAIDS device) is introduced to log into MAIDS. Once the password is validated, the desktop of the MAIDS device is shown in the RealVNC Viewer of the connecting computer, phone or tablet.

5.7 Sensor/effector code on repository

The Python source code for the MAIDS project is included in Appendix I of this document. The code file (maids_final_python_code_ceng355.py) is also found online through the MAIDS GitHub repository link:

1. https://github.com/srgawain2264/CENG317-MAIDS_PROJECT/blob/master/Python%20Code/maids_final_python_code_ceng355.py.

In order to maintain the project current, any and all future modifications to the code are promptly updated to the GitHub site.