



6 November 2014

Dr. Andrew Rawicz
School of Engineering Science
Simon Fraser University
Burnaby, BC, V5A 1S6

Re: ENSC 440 Design Specification for the Secure On-Site wireless package storage solution (SOSbox)

Included is SOS Technology's *Design Specification for the Secure On-Site Box (SOSbox)*. The information included within this document will outline design choices and technical specifications of the SOSbox. The document will discuss specific details regarding the design choices and implementation. The final implementation will follow the requirements and engineering standards outlined in the functional specification document.

Our goal at SOS Technologies is to create an efficient, reliable, secure, and technologically advanced solution for 21st century parcel delivery. Using the latest technologies and a well thought out top-down design, the product aims to fill a significant niche in the market for advanced parcel delivery.

SOS Technologies is comprised of five innovative engineering science students: Brett Hannigan, Dan Kikuchi, Herman Sagoo, Jackson Connolly, and Sutharsan Rajaratnam. These members collaborated using their unique skill-sets to design a functional and coherent product. Please feel free to contact me by email at srajar1@sfu.ca or by phone at 604-323-4647.

Sincerely,

Sutharsan Rajaratnam
President and CEO
Secure On-Site (SOS) Technologies

Enclosure: Design specification for the Secure On-Site Box.



DESIGN SPECIFICATION FOR THE SECURE ON-SITE BOX

SUTHARSAN RAJARATNAM
DAN KIKUCHI
BRETT HANNIGAN
HERMAN SAGOO
JACKSON CONNOLLY

Contact Person: Sutharsan Rajaratnam
srajara1@sfu.ca

Submitted to: Dr. Andrew Rawicz | ENSC 440W
Steve Whitmore | ENSC 305W
School of Engineering Science
Simon Fraser University

2014-11-06
Rev. 2

1 Executive Summary

Since its inception, the Internet has changed many things about how we interact with the world around us. In particular, online shopping has become the preferred purchasing method for many individuals and the meteoric rise of companies like EBay and Amazon is a testament to this change. This changing marketplace has also had the effect of spurring growth in traditional parcel service companies.

At Secure On-Site Technologies, we understand the substantial business opportunity in the aforementioned market segment. We have devised a product offering that will be a pioneer in modernizing the delivery industry. Our Secure-On Site box will enable individuals who frequently shop online to receive packages without having to be at home during business hours. By taking advantage of the user's Internet connection and a dedicated embedded hardware platform, we intend to build a solution that will save time and money for both the couriers and consumers in an evolving marketplace. The development of the SOSbox solution will go through three phases: proof-of-concept, prototype and production. We intend to introduce additional capabilities with each development phase and reduce cost where possible.

In this document, we have addressed all the design specifications of the product during each development phase. These specifications encompass all areas of the project including mechanical, hardware, software, and the physical enclosure. In addition, this document will describe the technologies we have elected to use in the SOSbox system as well as the reasoning behind these choices. It will also detail the test plan for the system for both subassemblies and the system as a whole.

Table of Contents

1	Executive Summary	ii
2	Introduction	1
2.1	Scope.....	1
2.2	Intended Audience.....	1
3	Specific Design.....	1
3.1	System	1
3.2	Physical.....	2
3.3	Mechanical.....	3
3.4	Electronics	5
3.5	Software Design.....	8
4	System Test Plan	11
4.1	Components	11
4.2	System	12
4.3	Software	12
5	Conclusion.....	12

List of Figures

Figure 1:	A dimensioned 3D model of the enclosure.....	3
Figure 2:	The lock-style solenoid.	4
Figure 3:	The drive schematic for the solenoid.	4
Figure 4:	A high-level system overview.	5
Figure 5:	The high-level diagram for the software architecture.	9
Figure 6:	A flowchart of the client software functionality.....	11

Glossary

ARM	advanced RISC machine, a processor architecture.
client	software that accesses a service made available by a server.
Darlington transistor	a pair of transistors whose behavior mimics a single transistor with a high current gain.
embedded system	a computer system-on-a-chip.
EMF	electromotive force, the voltage developed by any source of electrical energy such as a battery or dynamo.
fail secure	a failure occurs to the locked configuration
field of view	The extent of vision of an observer.
flyback diode	A diode that suppresses a sudden voltage spike when a supply is removed.
GPIO	general purpose input-output, a generic pin on an integrated circuit whose behavior, including whether it is an input or output pin, can be controlled by the user at run time.
GUI	graphical user interface.
I²C	inter-integrated circuit, a digital bidirectional communications protocol that uses two conductors.
IoT	“Internet of things”.
IP address	unique numeric address to identify devices on a network.
IR	electromagnetic radiation with a wavelength between 700 nm and 1 mm.
MJPEG	motion JPEG.
OS	operating system, the software that supports a computer's basic functions, such as scheduling tasks, executing applications, and controlling peripherals.
ping	a handshake message between a client and server.
PMMA	poly(methyl methacrylate), also known as acrylic or <i>plexiglass</i> .
PWM	pulse width modulation.
QR code	quick response code, a type of matrix barcode.
server	software capable of accepting requests from the client and giving responses accordingly.
SOShub	a scaled version of the SOSbox for implementation in community mailboxes.
TCP	a data transmission protocol that provides reliable, ordered and error-checked delivery over a network.
thread	the smallest sequence of programmed instructions that can be managed independently by an operating system.
UDP	user datagram protocol, a simple connectionless data transmission model with a minimum of protocol mechanism.
Video4Linux	a set of open-source Linux drivers for many web cameras.

2 Introduction

The SOSbox solution is an IoT-enabled secure parcel storage solution designed to benefit both the courier service and end customer by eliminating inefficiencies that exist within the parcel delivery process. Our solution aims to use the power of the Internet and embedded technology to provide real time, remote communication with the end customer. In this way, the user is able to receive parcels without being present on-site when the courier arrives. Additionally, the system will be designed with surveillance and physical theft deterrence capabilities for added peace of mind. Finally, the SOSbox solution will act as a demonstration platform for scaling up the underlying technology to replace the currently deployed community mailboxes with smart SOS Technologies systems. This document describes the technical details regarding the design of the SOSbox solution.

2.1 Scope

The goal of this document is to provide information regarding the design of the SOSbox system. The emphasis will be on the proof-of-concept version. The prototype and production versions will be discussed but such details are subject to change based on the proof-of-concept iteration. Furthermore, some details will be provided on the eventual development of a scaled up version of the SOSbox solution called the *SOShub* to illustrate the significant market potential of the underlying design and technology.

2.2 Intended Audience

The intended audience for this technical design specification is the members and advisors of SOS Technologies. Research and development engineers shall refer to this document as an overall guideline to ensure all design requirements are met. In particular, this document shall provide a starting point for further refinement of the proof-of-concept SOSbox solution and lay the design foundation for the production version.

3 Specific Design

3.1 System

The primary focus of this section will be to outline the physical, mechanical, electronics, and software design requirements for the proof-of-concept SOSbox. Additional design details will be included to illustrate a development road map that will ultimately lead to the production model of the SOSbox. We have

divided the design details into four subsections: physical, mechanical, electronics, and software. Each of the respective subsections will describe in detail the requirements for delivering a successful proof-of-concept device.

The SOSbox system centers around the BeagleBone Black microcontroller which controls all electronic components and communicates with the user's mobile device. This embedded system was chosen due to its ability to run Ubuntu Linux, large array of input-output interfaces, and low power consumption. Finally, its ARM Cortex-A8 processor has enough processing power to handle real time video streaming.

3.2 Physical

Because our company is aiming for a proof-of-concept design and subject to budget constraints, the SOSbox enclosure will be primarily constructed of off-the-shelf wood available at local lumber stores. Specifically, both the left and right frames are built around 2" by 2" wood pillars cut to the appropriate lengths. Both frames are adjoined by two single pieces of 2" by 2" wood of length 25" for the front and back. Upon completion of the frame, 1/2" plywood is fitted on the exterior and interior of these frames to protect the interior storage compartment from the elements.

Should the group gain access to the advanced mechanical labs at the British Columbia Institute of Technology, the exterior of the SOSbox will be covered with 22 gauge galvanized steel sheet metal. This will help with durability and waterproofing the enclosure. This design concept is displayed in Figure 1.

The top mounted body will house all associated electronics and the specified camera for QR Code and pattern recognition. It will be made of laser-cut PMMA sheets. This material was chosen because it is transparent (allowing the camera and LEDs to be located behind it), durable, easy to form, and easy to inscribe with status labels using the laser cutter.

A solenoid lock as described in Section 3.3.1 will be mounted inside the box adjacent to the door allowing access to be granted automatically.

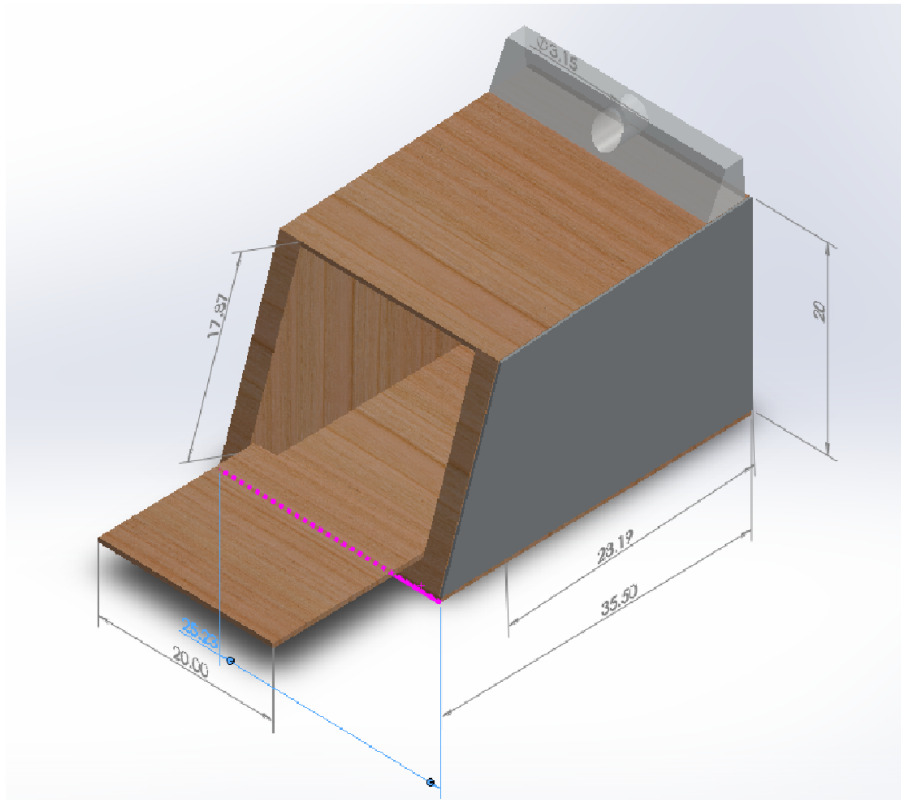


Figure 1: A dimensioned 3D model of the enclosure.

3.3 Mechanical

The mechanical design of the SOSbox parcel storage unit was heavily motivated by several factors such as security, reliability, weather resistance, ergonomics, and safety. As the product goes through the three product development phases it will be refined to improve on these factors.

For the proof-of-concept phase, several revisions were made to the functional specification design because of time and cost considerations. As a result, the scope of the proof-of-concept mechanical design outlined below has been limited to providing basic security, weather protection, and remote lock control.

3.3.1 Locking Mechanism

The locking mechanism consists of a lock-style solenoid depicted in Figure 2. The angled side of the solenoid allows the door to slip over it when closing. The flat side keeps the door locked. The lock is fail-secure; a spring extends the slug when not powered. The solenoid only consumes power to retract it momentarily while the door is being opened.

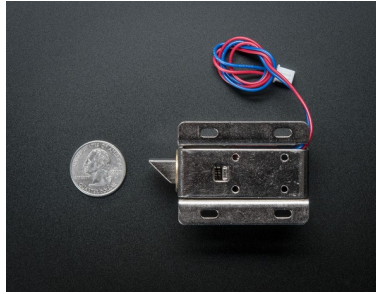


Figure 2: The lock-style solenoid.

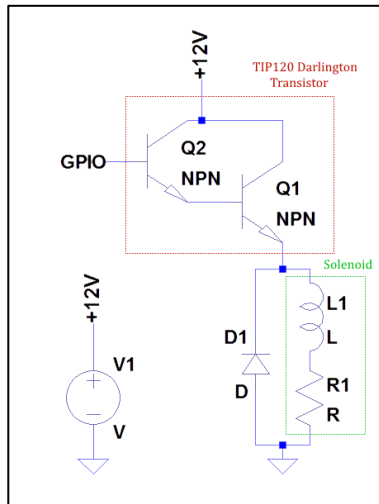


Figure 3: The drive schematic for the solenoid.

Referring to **Error! Reference source not found.**, the chosen solenoid consumes 650 mA at 12 V DC through a Darlington transistor where the base is connected to a GPIO pin on the BeagleBone embedded system. A flyback diode prevents back EMF voltage spikes from damaging the digital logic. Although the solenoid draws high power, it is actuated for less than ten seconds once per door opening-closing cycle so its impact on the SOSbox power consumption is very low.

The proof-of-concept device will use one lock-style solenoid mounted on one side of the door. The slug is made from mild steel with an approximate ultimate tensile strength of $S_T = 250 \text{ MPa}$ and a base cross-sectional area of $A = 1 \text{ cm}^2$. Using the approximate factor to convert from tensile to shear force of $\beta = 0.6$, an estimation of the shear strength of the lock is calculated in Equation 1:

Equation 1: Estimation of shear strength of the lock-style solenoid.

$$S_s = \beta S_T \cdot A$$

$$S_s \approx 0.6 \cdot 250 \times 10^6 \frac{\text{N}}{\text{m}^2} \cdot 1 \times 10^{-4} \text{m}^2 = 15 \text{ kN}$$

The shear strength of 15 kN is considered adequate for the purposes of the proof-of-concept device because it exceeds what force a person could apply to the door alone. Also, the lock receptacle and door will likely fail before this threshold. By designing the door to close flush with the enclosure, the force that could be applied on the lock by a person using tools can be mitigated.

3.4 Electronics

This section of the document provides an overview of the hardware used in the SOSbox parcel storage unit. A high-level system block diagram of the embedded platform and connected peripherals is presented in Figure 4. Furthermore, the subsections below will provide additional details on the digital sensors, attached peripherals, motor control, power supply, and user interface of the SOSbox parcel storage unit.

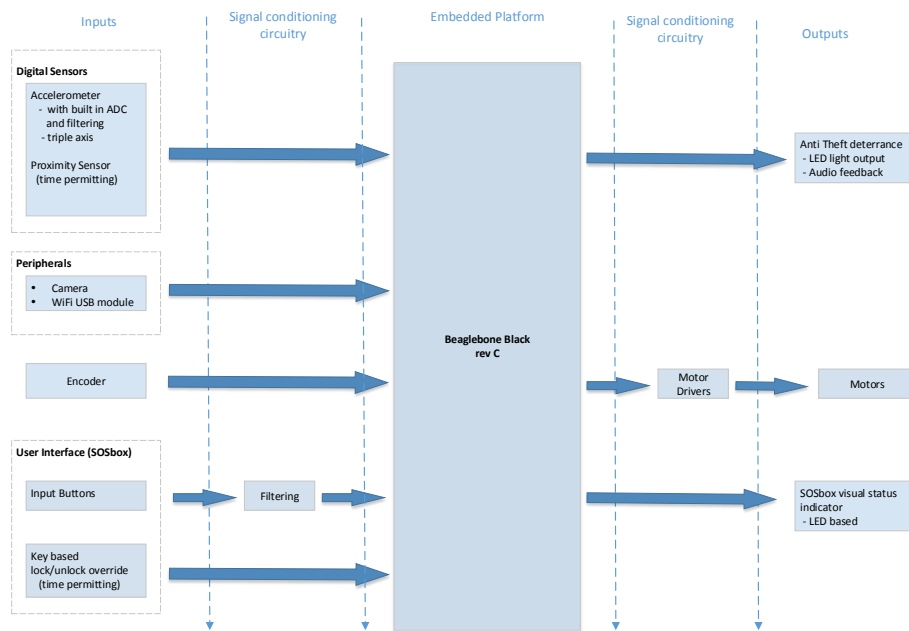


Figure 4: A high-level system overview.

3.4.1 Surveillance

In the proof-of-concept phase, the surveillance system will use an off the shelf USB web camera. Our focus is not on the camera hardware but rather the software development necessary for pattern recognition and video streaming. The camera will be mounted in a fixed position on the upper part of SOSbox unit to allow maximum field of view. It will connect via USB directly to the BeagleBone, as illustrated in , for both power and data transmission.

Video capture capabilities are required for the streaming and processing functions done in software. An off-the-shelf Microsoft HD USB webcam is used for capture. This camera was chosen because it is compatible with Video4Linux drivers, is high resolution, has autofocus, and has hardware video compression to reduce the load on the BeagleBone embedded system. The compression uses the motion JPEG (MJPEG) encoding scheme. MJPEG is intraframe compression where each frame does not depend on other frames for encoding and decoding. This means frames can be streamed or dropped in a simpler way.

The autofocus capability is important for detecting QR codes on packages automatically. The same camera will stream the video feed and capture package QR codes. Future expandability includes a built-in microphone for audio streaming and full 1080p HD resolution video.

A custom designed ring of 20 infrared LEDs will surround the webcam and allow imaging in low light conditions. Each LED consumes 100 mA and has a viewing angle of 20°. If this is not adequate, the camera will have to be modified to remove its IR filter. Another Darlington transistor will drive the LED array from the 5 V power supply as it consumes 2.8 W of electrical power.

For the prototype and production phases, we will design custom camera circuitry to enable enhanced low light performance and motorized multi-axis pivoting capabilities. This will allow us to introduce new features while optimizing space and housing requirements within the SOSbox unit.

3.4.2 Sensors

Despite the metal enclosure providing a level of mechanical theft resistance, an accelerometer will be used to provide motion data for the on board theft deterrence system. A warning alarm will alert the user if a theft is in progress. The Analog Devices ADXL345 3-axis digital accelerometer was chosen over other options because of its low cost, multiple communications interfaces, and built-in threshold motion detection.

The accelerometer will be connected to the BeagleBone embedded system using the I²C protocol to reduce the number of data wires. The ADXL345's built-in threshold detection uses a register in the accelerometer that can be set to a predetermined value. When this acceleration threshold has been exceeded, the accelerometer IC will send an interrupt to the BeagleBone and trigger the notification and audible alarm. In this way, the BeagleBone system does not have to constantly sample the accelerometer output and can save resources for video streaming and QR code detection.

For the production device, a passive infrared proximity sensor will be used to implement active automatic surveillance efficiently. This will give the SOSbox situational power management capabilities and enable the unit to reduce power consumption when no activity is present in its vicinity. Climate control capabilities will be implemented using temperature and humidity sensors in future iterations.

3.4.3 Power supply

The embedded platform will be powered through a 12 V, 5 A AC to DC mains adapter. The lock solenoid and door actuator motor can run directly off the adapter's DC output. The remaining electronics require 5 V DC provided by a Texas Instruments LM2679 step-down switching regulator.

For the future development phases, the SOSbox unit will incorporate an independent self-contained back up power supply to power the unit for a minimum of 1 month. Possible on board solar recharging will be explored to determine feasibility and cost effectiveness.

3.4.4 User Interface (SOSbox parcel storage unit)

The unit will feature a momentary push button connected directly to the embedded platform's digital input. The function of the push button is to wake up the SOSbox unit at time of parcel delivery. A short delay will be introduced in software to eliminate problems with switch bouncing. In the absence of motion detection, a physical method to activate the unit is required. Additionally, LED lights will convey SOSbox states such as ON, OFF, ERROR, PLACE PARCEL, and PARCEL STORED.

The interface will be completely revamped for further development phases. A touch screen interface will be used to provide superior user input capabilities and enhance information output. Furthermore, audio capabilities along with Internet connectivity will enable real time two-way voice and video communication.

3.4.5 Motor control

For the proof-of-concept device, the door will be mechanically actuated to remove user involvement from closing and locking the door. This improves security by reducing human factors. A stepper motor has been selected to perform this function because it has high torque to seal the door correctly, low cost, and no feedback control required.

The stepper motor will be run off the 12 V bus using the Texas Instruments L293D H-bridge driver IC. The BeagleBone's PWM outputs can be used to modulate the motor speed and momentary limit switches at each end of the door's stroke will stop the motor when the door is fully closed or opened. Further experiments are necessary to determine the force required to open and close the door, less than 10 N is expected. Deviations from expected force requirements will be made up by adjusting the gear ratio.

3.5 Software Design

The proof-of-concept version of the software will consist of a server capable of controlling all electronic components and a client application capable of communicating with the server.

The prototype version will have an improved user interface, as well as server support for multiple users and reduced data transmission requirements. For the production model, the server will have improved secure communications and user verification along with support for clients to control multiple SOSboxes.

To simplify development and testing, the software has been broadly divided into four separate areas. The server software, hardware drivers, and streaming thread are all executed on the on-board microprocessor in the physical box. The figure **Error! Reference source not found.** depicts the full system software and its interactions throughout the product. The mobile application will run on a compatible mobile device. The server and client applications will be designed to run in Linux and Android OS environments respectively. The design uses a Wi-Fi module to connect the BeagleBone, to avoid the need for an Ethernet cable to connect to the SOSbox.

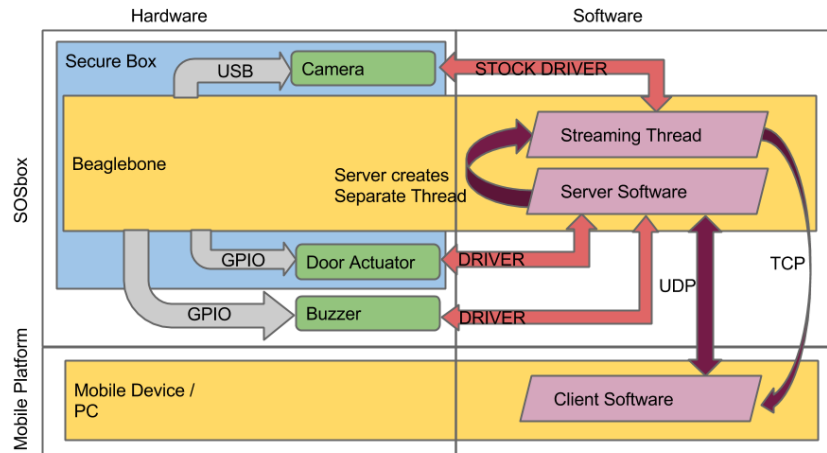


Figure 5: The high-level diagram for the software architecture.

3.5.1 Server Overview

The server software will be running on the BeagleBone microprocessor in the SOSbox in an Ubuntu Linux environment. This was selected for three reasons: driver support (for the Wi-Fi module and camera components), development compatibility, and community support. For the server we chose the programming language Python, due to widely available BeagleBone software packages, community support and code portability. The client-server connections will use an Internet connection and the UDP protocol (as implemented in the Python native socket library) for low-level communication.

The server has a graphical user interface (created using the *tkinter* Python library) for testing purposes which passes commands to the main thread running in parallel. This thread refreshes at regular intervals and performs the functions outlined below.

The server maintains a list of active clients (clients who have pinged the server recently) and accepts data packets from them. These queries are scheduled to be executed and once they are completed, confirmation is sent to the client. These tasks include running hardware drivers for door control, reading states from sensors and the buzzer, and creating and destroying parallel streaming processes.

3.5.1.1 Streaming Thread

The streaming thread uses third party open source software (*ffmpeg* or *live555* libraries) to establish streaming connections to clients when they are requested.

In addition, the streaming thread will use the camera component and third party open source software (*zbar* library) to recognize QR code patterns in the video stream. To avoid overloading the BeagleBone microprocessor the QR code recognition will only run when at least one valid QR code is set. Similarly, the streaming video connection will only operate when requested, and the QR recognition will be disabled when it is active.

3.5.1.2 Hardware Drivers

The server will interface with the SOSbox via custom drivers written by the hardware team. The hardware drivers will actuate the electrically controlled functions of the SOSbox detailed in the mechanical and electronics sections of this document, as well as acquiring data from the accelerometer and other sensors.

3.5.2 Client Requirements

The client side encompasses the application on the user's mobile phone and its interactions with the server, which mediates all the user's input. The production version of the application will be available for all mobile platforms in their respective app stores, however for the prototype we will be using an Android Java application installed on the mobile phone beforehand.

The design of the application is straightforward. Upon loading, it will display a GUI with several options shown in the flow chart in Figure 6. The options will be cleanly laid out top to bottom in an aesthetically pleasing manner. The options will include but are not limited to: a video display, a manual lock and unlock, as well as preferences. The client software will be kept minimalistic to lessen the burden on whatever device the user may have. This will allow users with even simple legacy devices to use the SOSbox. To set up the client, the IP address must be set to that of the SOSbox. This can be changed through preferences manually in the case that the user wants to track a different SOSbox.

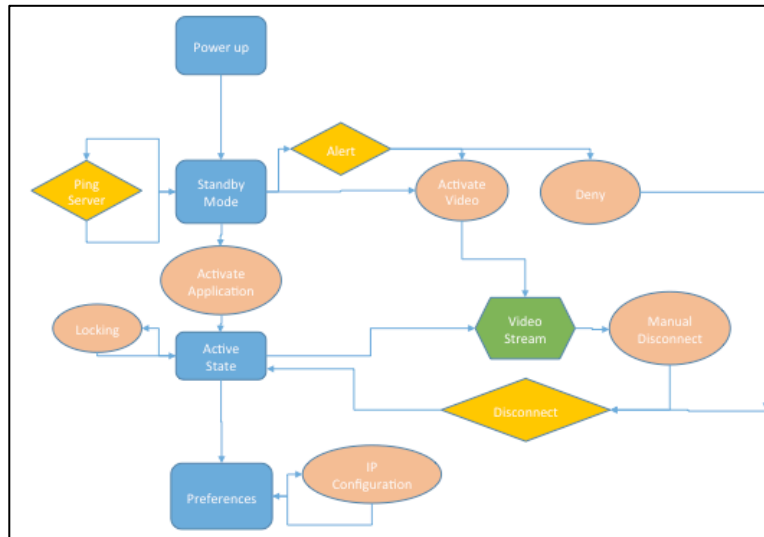


Figure 6: A flowchart of the client software functionality.

The application will run a low-impact process in the background to periodically ping the server to check if the server is still online, and check for new data packets from the server.

The video stream will use a direct TCP connection. It will be displayed in a separate screen when the stream is manually opened and confirmed by a server packet. The stream will continue to run until the user exits the stream at which point the process will continue to run, but not the stream. As discussed in Section 3.4.1, the stream will have a low VGA resolution for the prototype. The options to toggle on the lock will be displayed below the video. The information from a button press will be sent to the hardware in parallel with the stream using UDP.

4 System Test Plan

4.1 Components

Testing individual components prior to integration into the entire system is critical to reduce debugging time and increase system reliability. Standard instruments will be used to bench test all electronic components and subassemblies as they arrive. The output of the accelerometer can be calibrated using a PC. Afterwards, verification will be done to ensure the entire electronics system meets the functional specifications.

4.2 System

Once all software and hardware components are tested, the system can be assembled in the enclosure. The enclosure will be tested for its performance in a wet outdoor environment. The door mechanism and seal can be tested in a similar way. The enclosure's attenuation of the Wi-Fi signal can be tested using software on the embedded system and a reference PC. Finally, test subjects can be recruited to evaluate the interface layout.

4.3 Software

Each of the software functions will first be tested independently and then there will be custom benchmark tests for the hardware drivers and streaming thread. The benchmark tests allow sample rates to be optimized.

The client application must be tested in conjunction with the server, however, the server will also be tested using a desktop test program to ensure it functions correctly. The streaming thread will also be tested using a desktop test program, and the QR code reading will be tested using custom printed codes.

We will be attempting for response time within one second of an event, and less than 5% processing capability used when idle. The streaming capabilities must function without reducing server response time.

5 Conclusion

The proposed design specifications for the SOSbox solution illustrate the group's intent to meet the functional specification laid out for the proof-of-concept phase. As with any product development, the initial goal of a proof-of-concept design is to demonstrate the technical viability of the product. Subsequent product development phases will substantially improve on the initial design. Design decisions provide scalability of the SOSbox system with the intended goal of diversifying the courier service industry.