# 2. Approach

The Quicket System is designed to improve an area of daily life everyone understands: waiting in line. The system accomplishes this using a thermal camera to detect the number of people entering the gate and an UHF RFID reader to scan tickets. The inputs from the thermal camera and RFID reader are then passed into the Raspberry Pi, where the tickets are cross checked with a database. After this, the correct tickets scanned are compared with the number of people detected by the thermal camera. The Raspberry Pi will indicate the system passes until there is a mismatch between the two inputs at which point it will switch to a different output: calling for human intervention.

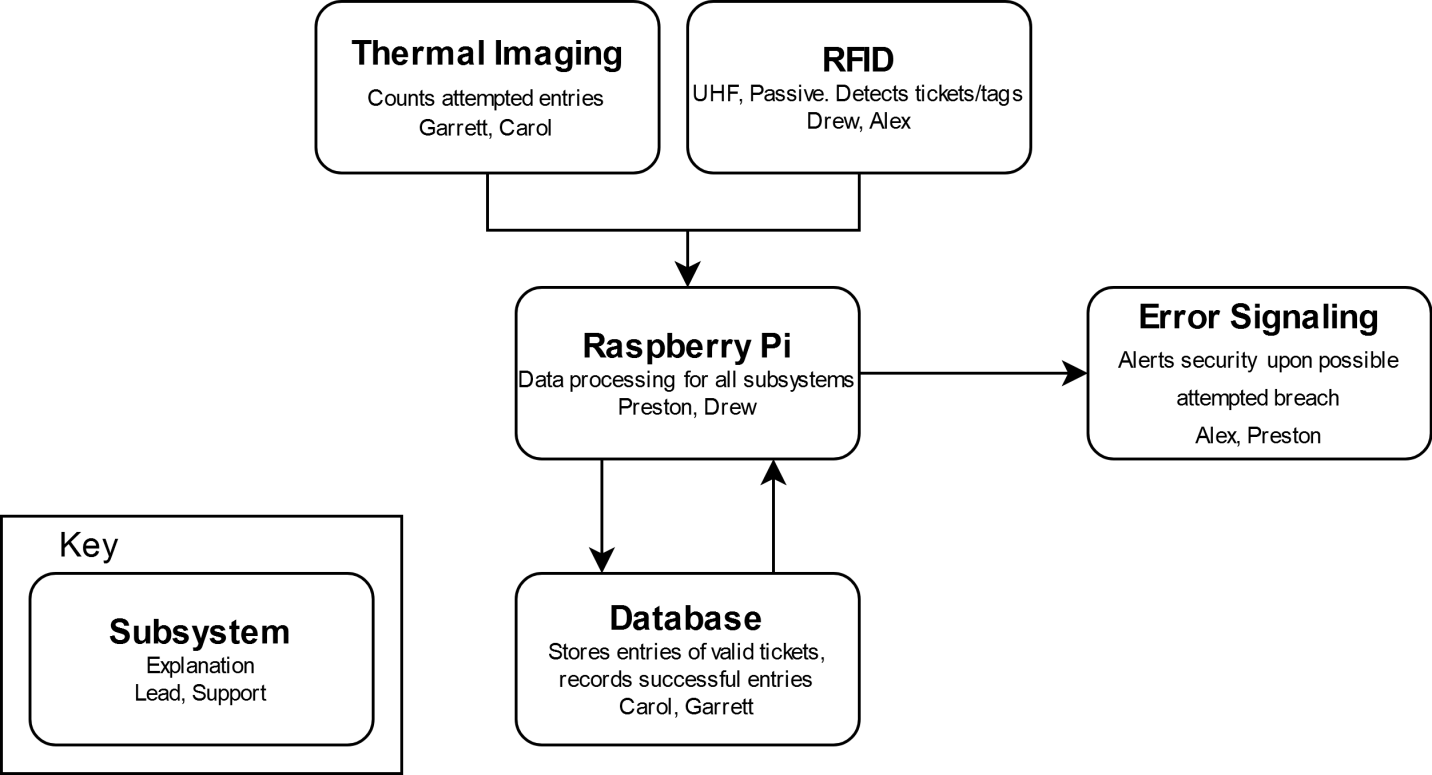


Figure 2a: The Quicket System Subsystems

## 2.1 Hardware

The Quicket System relies on four major hardware subsections: the ticket reader subsystem, the controller subsystem, the security subsystem, and the error signaling subsystem. These subsystems are further detailed in the following sections.

**2.1.1 Ticket Reader**

The first subsystem involves the ticket reader itself including an emphasis on what type of signals are utilized, the model of the reader, and its antenna characteristics. This system is responsible for detecting and processing which tags pass through the threshold and passing that data to the system controller.

**2.1.1.1 Signal Type**

Numerous types of signals are used throughout daily life, and several of them either already handle ticket processing or hold such potential. After initial research, the team selected four signal types, shown in Table 2.1a, for consideration in The Quicket System.

Table 2.1a - Signal Types

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Technology | UHF RFID | Bluetooth | NFC | UWB |
| 2 Meter Range | Yes | Yes | No | Yes |
| Active/Passive | Passive | Active | Passive | Active |
| Tag Handling | Multiple | Multiple | Single | Multiple |

RFID is a common technology based off the principle of a reader that interacts with tags over a variety of radio wave frequencies. These frequencies are broken into three main categories: low frequency (30 KHz to 300 KHz), high frequency (300 MHz to 30MHz), and ultra-high frequency (300 MH to 3 GHz) [6]. The general idea is that a lower frequency means smaller read distance and slower data transfer but less interference from metals and liquids. The crucial differences in these separate categories stem from the available read range. The low frequency signals offer the shortest read range, almost exclusively under 10 cm. This range increases to a full meter with high frequency RFID. However, ultra-high frequency signals greatly surpass the other two types in read range. It routinely creates read ranges of 10 meters up to 100 meters with the proper tags and antennas [6]. Given the differences between the three categories, it makes sense for the design team to only consider ultra-high frequency RFID moving forward.

Within UHF RFID systems, there are two major types of tags. Passive tags use the energy transmitted by the reader to power their own internals to radiate a signal back. Active tags contain a built-in power supply that emits its own signal for the reader to pick up. Active tags contribute to a much longer read range than passive tags, but they are more expensive to produce and have a shelf life dependent on their power source. Additionally, passive tags still offer a range that easily covers the required 2-meter range of the system. As such, passive UHF tags offer the best solution from RFID options for The Quicket System.

Bluetooth signals mainly focus on linking communications between two devices. This idea could be applied for ticketing measures, especially when paired with a Bluetooth beacon. Bluetooth beacons constantly emit signals that are picked up by Bluetooth compatible devices such as smart phones [7]. These signals contain some sort of “push” data that prompts the user of the device to allow a certain action. A possible implementation for ticketing involves the user storing tickets on the device. As the device nears a Bluetooth beacon, the user receives a prompt to “activate” the valid ticket. A system of beacons triangulates the device and determines when it passes through the gate. However, this type of implementation heavily relies on user input and most likely would include some sort of app development.

NFC technology is already prevalent for ticketing as well as a variety of other uses. It focuses on linking compatible devices together similarly to Bluetooth but provides a much more secure connection. It is also a passive technology that uses magnetic induction to provide power. However, this secure connection comes at the expense of the signal range. The max range of traditional NFC devices is around 0.1 meters [8].

Ultra-wide band (UWB) technology is another option that has seen increased popularity over the past few years, even being integrated into the newest models of Apple’s iPhones. UWB operates similarly to the other options by transmitting and decoding pulses of radio waves. The two biggest advantages of UWB based system are that it is resistant to interference and that it enables accurate distance measuring. These measuring capabilities are due to UWB’s Time of Flight (ToF) basis, a timing system that records how long it takes to send and receive signals from one device to another [9]. However, these tags and devices require active components dependent on power supplies.

Given the current ticketing options available on the market, the design team agreed to focus on two main attributes for The Quicket System: multi ticket processing and a passive approach that removes user dependability. As such, Bluetooth is not an appropriate signal base for the design. Furthermore, NFC ticketing kiosks already exist, but this technology does not currently enable a read range to cover the needs of the system. This leaves the final decision between UHF RFID and UWB signals. While both meet the system requirements, there is a significant difference in the options and price range of UHF RFID tags compared to UWB tags. While the distance measuring aspect of UWB could be impactful for error detection, it does not outweigh the system flexibility available through UHF RFID.

**2.1.1.2 RFID Reader**

Given the information above, the team researched various UHF RFID readers. The team is not considering readers over $300 due to budget constraints and most of these being overly equipped for The Quicket System’s needs. The team narrowed down options to the three readers in Table 2.1b.

Table 2.1b - RFID Reader Options

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Fonkan FM-503 [10] | Sparkfun M6E Nano [11] | LPSECURITY UHF-105 [12] |
| Price | $85 | $225 | $92 |
| Max Range | 4 m | 4.9 m | 7 m |
| RF output power | 0-25dBm | 0-27dBm | 0-30dBm |
| Antenna | Yes | Yes | Yes |
| Gain/Polarization | 5.5dBi, Circular | Not specified | 8dBi, circular |
| Range with Antenna | 3 m | 0.6 m | 5 m |

The Fonkan FM-503 reader offers a middle-ground option, providing ample range at a price within the team’s budget. Additionally, it gives the ability to adjust output power and comes with an antenna that provides enough range for the system requirements. Overall, this reader meets all The Quicket System’s constraints at a respectable price.

The Sparkfun M6E Nano offers the best processing power at the highest price point of the three options. The read range with the integrated antenna does not meet the design requirements which would require an additional antenna. Given the steep price point and the need for an additional antenna, this reader was not chosen.

The LPSECURITY UHF-105 offers the strongest signal of the three options and meets the system requirements. However, too much strength causes problems within the system as it is now likely to detect tickets outside the proper range, increasing the likelihood of false error reports. This signal strength also nears the allowed limit without offering many benefits, and, as such, the reader is not a good fit for the system.

**2.1.2 Antenna**

A crucial component for the ticketing system comes from the RFID antenna. The antenna radiates the signal supplied by the reader and picks up the backscatter emitted from the UHF tags. One of the largest impacts from the antenna depends on its polarization. There are two major types of polarization: linear and circular. Linearly polarized antennas produce signal waves that oscillate within one plane. Signals from circularly polarized antennas oscillate in two, perpendicular planes. Linearly polarized antennas typically possess longer ranges but are more susceptible to tag orientation problems. Circularly polarized antennas show shorter read ranges but compensate by allowing higher degrees of error in tag orientation.

Another important aspect lies in the radiation pattern of the antenna and its beam width. Combined with polarization, this trait greatly affects the effective zone of the antenna. Too narrow of a beam width means the antenna misses tags that pass on the edge of coverage while too large of a beam width means the system picks up excessive or improper tags. The FM-503 antenna is circularly polarized and possess an effective radiation and beam width. As such, the supplied antenna sufficiently covers the required read range while providing a margin of error for tag orientation. The corresponding antenna radiation plot is shown in Figure 2.1a.

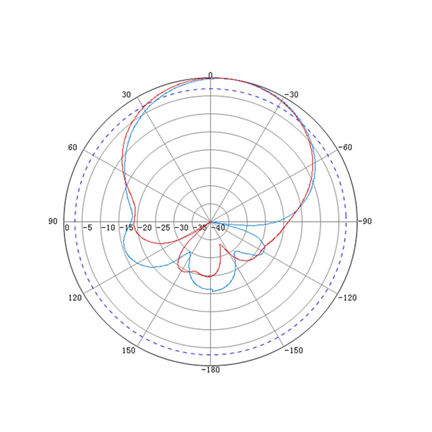


Figure 2.1a: Antenna Radiation Plot [13]

**2.1.3 System Controller**

The Quicket System must communicate and process information relatively quickly to meet technical design constraints. When considering options for the primary computer, three major boards fulfill The Quicket System’s needs: Raspberry Pi, Arduino, and BeagleBone.

Table 2.1c -

|  |  |  |  |
| --- | --- | --- | --- |
| **Controller/Computer** | **Processor** | **RAM** | **Output** |
| Raspberry Pi 4 B | 1.5 GHz, 64-bit | 4 GB | 2x Micro HDMI, 2x USB 2.0, 2x USB 3.0, Ethernet, MIPI DSI display port, MIPI CSI camera port, 1x 40 pin GPIO header |
| BeagleBone Black | 1 GHz, 32-bit | 512 MB, 4 GB flash | Ethernet, USB 2.0, HDMI, 2x 46 pin headers |
| Arduino Uno | 16 MHz, 8-bit | 2 KB, 32 KB flash | 14 Digital I/O pins (6 PWM, 6 analog) |

Arduino boards, while a less expensive than the latest Raspberry Pi, lack some hardware specifications that make it less efficient within the scope of The Quicket System. Arduinos used for RFID scanning would be limited in networking capabilities and processing speeds. An Arduino uses an 8-bit microcontroller that better suits larger amounts of control I/O [14]. While the board can handle larger currents, these are not necessary for The Quicket System’s applications. Most importantly, the Arduino only has 2 kilobytes of RAM available, making it more of a small-scale controller than a multi-functional computer [14].

The chosen computer or microprocessor for The Quicket System provides reliable network communication with a server database for tag identification. To relay this information between the server and the computer the RAM and CPU clock speeds must be substantial. The Arduino Uno does not come with network cards by default and needs more adaptation.

Most applications that suit the Arduino pertain to specific subsystems or sensors that are measured or operated systematically. Applications for PLCs and simpler control logic conform better to the Arduino’s designed purpose.

The BeagleBone and the Raspberry Pi 4 have similar hardware specifications. The BeagleBone Black utilizes a 1 GHz, 32-bit processor and 512 MB of RAM with 4 GB of flash storage [15]. These features, along with its 3D graphics accelerator and dual 32-bit microcontrollers, make the BeagleBone capable of delivering similar performance to the Raspberry Pi [15]. Though the Raspberry Pi 4 has higher clock speed, RAM, and overall output, the BeagleBone would meet The Quicket System design constraints.

Though comparable in performance, the major drawbacks of the BeagleBone are lack of community support and hardware integration. These aspects combined with the slight dip in performance make the Raspberry Pi more useful for testing and design implementation. Additionally, the BeagleBone does not offer the same out-of-the-box integration with other components chosen for the thermal imaging and RFID reader subsystems as the Raspberry Pi.

The Raspberry Pi provides more robust technical specifications. With a 64-bit processor and up to 8 GB of RAM on the latest Raspberry Pi 4 B, it can handle more processes at one time than the Arduino. A clock speed of 1.5 GHz and a quad-core CPU [16], the Pi’s CPU surpasses the Arduino’s 16MHz CPU by around 837% and the BeagleBone’s 1 GHz speed by around 50%. This increase in clock speed and available RAM allows the Pi to function with more demanding computational loads. The additional computational power of the Raspberry Pi allows for easier full-scale integration with complex programs needed for the database interaction and thermal camera subsystems.

When considering the interfacing methods between the primary computer and the peripherals, there may be additional hardware required to split or join cabling to the various hardware components. The chosen computer handles multiple digital and analog inputs, higher voltage outputs, as well as different protocols for the thermal camera and the RFID reader. Each of the 40 GPIO pins on the Raspberry Pi 4 can allow for up to 3.3V and a max current of 16mA output. This can power LED and control outputs, in addition to the thermal camera operation. The Raspberry Pi also offers accessible display outputs for two HDMI ports and MIPI camera/display ports [16], allowing for the use of a touchscreen/monitor display for real-time troubleshooting.

The Raspberry Pi offers major advantages by working more independently of connected devices. Some hardware components for both the RFID reader/antenna subsystem and the thermal camera subsystem can be programmed in multiple languages to communicate with the Raspberry Pi, whereas, the Arduino is primarily coded in C++ [17]. The added configurability of the Raspberry Pi allows for handling unforeseen errors or behaviors. The system can be adapted to suit the needs problems with a wide variety of user-made content.

Another consideration of the controller choice was economic cost. Having already acquired the Raspberry Pi 4 B and appropriate power connections, it would not affect the budget. Any alternatives would reduce available funds needed for higher quality thermal cameras and UHF RFID reader systems.

Overall, the Raspberry Pi’s availability at no cost, combined with its superior hardware specifications and component compatibility, made it the best choice for The Quicket System’s primary computer.

**2.1.4 Error Signaling**

The Quicket System’s final stage is the error signaling subsystem, which calls for human intervention upon a disagreement between the number of people detected by the thermal camera and the number of correct tickets detected by the RFID system.

The system operates on a pass until fail basis. The green LED signaling there are no problems at the gate will remain on until there is a difference between the number of people entering and the number of correct tickets scanned. The Raspberry Pi will switch the output from one pin to another when there is an error in the scanning process. The green LED indicating there is no problem at the gate is located on pin 1 of the Raspberry Pi and the red LED indicating there is a mismatch between the number of people and tickets is located on pin 17 of the Raspberry Pi.

The pins of the Raspberry Pi supply 3.3V at 16mA maximum current. The LEDs were chosen to operate at 2V at a maximum current of 20mA because they are rated above the maximum current being provided and operate below the maximum voltage supplied. The LEDs need an 81.25 ohm (1) resistor to provide a 1.3V drop between the output of the Raspberry Pi and the LED. 100 ohm resistors are used due to tolerance and availability.

 (1)

This particular approach was chosen because the Raspberry Pi is already an integral part in determining the passing or failing when users enter the gate. The internal functionality of the Raspberry Pi is exploited in order to maintain simplicity of the system; rather than utilizing external switches such as a pFET, the system utilizes what is already available.

The 4 possible states of the system are listed below in Table 2.1d.

Table 2.1d - Error Signaling States

|  |  |
| --- | --- |
| **Inputs** | **Outputs** |
| T = P = 0 | Pass |
| T > P | Fail |
| T < P | Fail |
| T = P | Pass |

T – Tickets

P – People

The system only indicates an error and calls for human intervention when the number of people entering the gate do not match the number of correct tickets scanned by the RFID.

**2.1.5 Thermal Camera**

The Quicket System uses two separate data streams for admitting users. This, in and of itself, is not unique to The Quicket System. Almost every conventional ticketing systems use secondary “data stream,” of sorts, usually in the form of human intervention rather than a literal stream of data. The Quicket System differs in that the secondary data stream is a from a thermal camera.

The Quicket System utilizes thermal imagery to count the number of people attempting entry and compare that to those who have been authorized entry. If there is a discrepancy between the number of tags detected by the RFID subsystem and the number of people counted by the thermal imaging subsystem, then an unauthorized entry has occurred.

The three thermal cameras considered for their applicability to these general needs are listed and compared below in Table 2.1d.

Table 2.1e - Thermal Camera Options

|  |  |  |  |
| --- | --- | --- | --- |
| Model | MLX90640 Thermal Camera Breakout with PIM366 | MLX90640 Thermal Camera Breakout with PIM365 | FLIR ONE Pro |
| Price | $60.84 | $60.84 | Available through MSU |
| Lens Angle | 110°x75° | 55°x75° | 55°x 45° |
| Thermal Resolution | 32x24 | 32x24 | 160x120 |
| Frame Rate | 64 FPS | 64 FPS | 8.7 FPS |
| GPIO | Yes | Yes | No |
| Compatible with RPi | Yes | Yes | No |

The Quicket System makes use of the MLX90640 Thermal Camera Breakout which uses the PIM366 thermal camera. The PIM366 has a resolution of 32x24, and a wide-angle (110°x75°) lens [18]. While this resolution is low, the MLX90640 fits the needs of The Quicket System.

The temperature detection capabilities of this camera exceed the needs of the thermal imagery processing software, at a range of -30°C to 400°C, an accuracy 1°C, all while maintaining a framerate of 64FPS.

The MLX90640 natively supports the Raspberry Pi, utilizing the I2C protocol for communication over GPIO pins via a utility library.

The wide-angle variant (PIM366) of this camera is used so the system can detect people attempting to enter the RFID reader’s range. The standard, non-wide-angle variant (PIM365) would only detect those directly below the camera and therefore already within the range of the RFID reader [19].

The FLIR ONE Pro is the most powerful camera in the comparison and is provided at no cost to the team. However, this camera does not fit the needs of The Quicket System. The FLIR ONE Pro provides no GPIO compatibility and is only operable in conjunction with an Android smartphone [20]. If efforts were made to make the FLIR ONE Pro compatible with a Raspberry Pi, it would still not fulfil the needs of The Quicket System. The FLIR ONE Pro does not have a lens angle wide enough [20] to detect approaching people. Additionally, although the image fidelity of the FLIR ONE Pro [20] far surpasses that of the other thermal cameras considered, the frame rate [20] is not sufficient for the high-fidelity motion-tracking needs of The Quicket System. If the time difference between each frame is too high, the software may not be able to distinguish between an already existing person moving and a new person entering the thermal camera’s view.

## 2.2 Software

The Quicket System has two software subsystems. The thermal imaging subsystem sends the number of people attempting to pass through the RFID reader’s range to the database interaction subsystem. The database interaction subsystem checks data received from the RFID scanner against both the database of valid tickets and against data received from the thermal imaging subsystem. If there is a discrepancy between the number of people detected and the number of tickets, an error is raised and reflected in the error signaling subsystem

**2.2.1 Database Interaction**

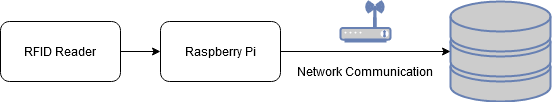
The Quicket System uses MariaDB (a fork of MySQL) to store the IDs associated with each RFID ticket. MariaDB was chosen because it is an easily accessible, non-proprietary database software that can be hosted on a consumer computer. This feature makes it straightforward to test the implementation of the microcontroller-database network communication and will reduce the costs associated with the project. Additional benefits offered by MariaDB include the extensive documentation provided for the system, non-complex vertical scalability, and its ability to add security to the system through data encryption. [21]

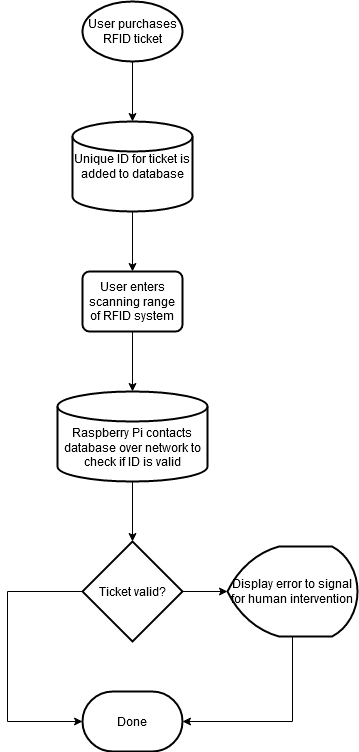
PostgreSQL is an open-source, MySQL-based system with support for a wide variety of custom data types such as JSON and XML. However, this system was not chosen because of its poor documentation, where many details are either inconsistent or missing altogether. The system also suffers from a lack of tools that show the state of the database, which would make it difficult to keep the database stable. [21]

MySQL was the third database management system considered for the project. The system was considered because it is the most popular relational database system and works well with a wide variety of computer systems, making implementation into most forms of servers straightforward. MySQL is also easy to learn with simple syntax. MySQL was not chosen for The Quicket System because it does not feature easily implemented scalability, imposes additional costs for the project if features not offered by the free edition were required, and would not work well with a large volume of data.

When a ticket is purchased, the unique identifier associated with it is stored in the MariaDB database. This unique identifier is programmed into the RFID tag held by the end user. When the end user enters scanning range of the RFID reader in the gate, the reader sends the recorded identifier to the Raspberry Pi. The Raspberry Pi transmits this identifier over the network to the database, which it is then checked against. If the identifier is found in the database, then the ticket is valid and no intervention is required; however, if the identifier is not found in the database, the ticket is not valid. An “OK” or “not OK” flag will be returned by the database to the Raspberry Pi. If an “OK” flag is received, then the Raspberry Pi will not alter the always-on green “OK” LED. If a “not OK” flag is received, then a signal is sent to the LED system that will illuminate a red light. This red light will signal the need for human intervention.

The system will not cease the scanning of RFID tickets while in the “not OK” state. Therefore, communication between the Raspberry Pi and the database will continue until the system is powered off.

  
**Figure 2.2a: Overview of Reader/Database Interaction**

  
**Figure 2.2b: Tag Reading and ID Checking Logic Overview**

**2.2.2 Thermal Imaging**

Video taken from the thermal imaging hardware is fed into a designed program that detects the number of people attempting to pass through the detection area. This program leverages several libraries in a Python environment to this end: MLX90640, a utility library for the system’s thermal camera (provided by the manufacturer); OpenCV, a computer vision and machine learning library; NumPy, a scientific computing library; and imutils, a convenience library that streamlines parts of the OpenCV and NumPy pipeline.

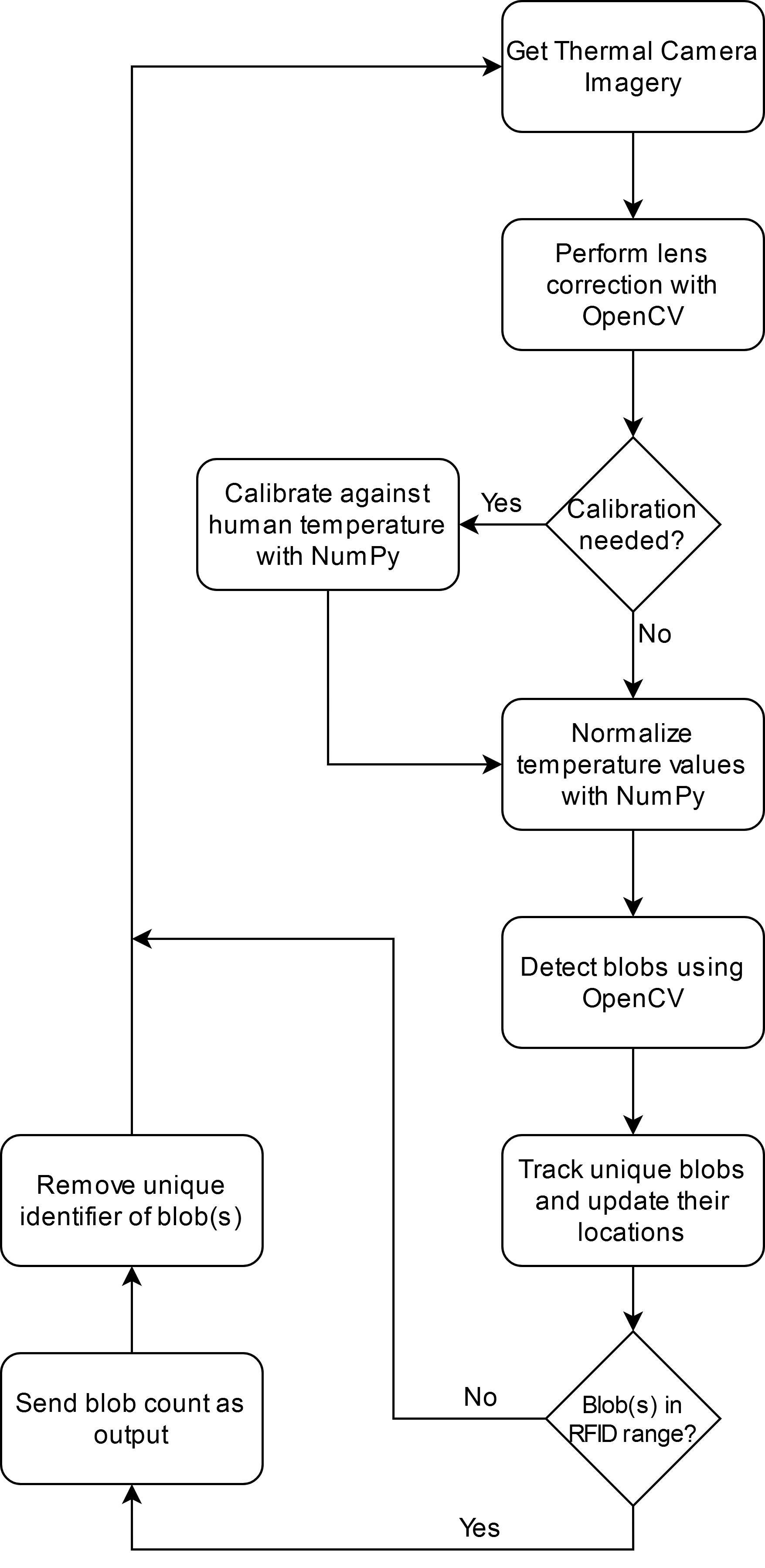
These libraries are used to render counts of people currently passing through and those approaching the RFID reader’s range. The count of people passing through the RFID reader’s range is then compared against the number of tags detected by the RFID reader. The count of people approaching but outside the RFID reader’s range is used later in the program and is detailed further on in this section.

The first step in the process performed by the thermal imaging subsystem is image extraction and correction. Images extracted from the thermal camera’s video stream via the MLX90640 library are processed and corrected for various environmental and technical factors, such as fisheye lens distortion, sensor inaccuracies, and thermal noise. OpenCV is used to correct fisheye lens distortion, and NumPy is used for output calibration and thermal noise correction. The thermal output is calibrated such that the average human temperature is close to 37.0 °C (98.6 °F). This temperature is chosen because it serves as a dependable ground truth against which to calibrate. Finally, the matrix of temperature information is normalized using NumPy.

OpenCV’s blob detection methods are used to identify people using the image and temperature data to find the number of people. By comparing the blobs detected from one frame of video to that of the next, the velocity of each person can be detected. With this information, each person is tracked individually and is assigned a unique identifier until they leave the thermal camera’s range.

The position of each person is used to determine their distance from the RFID reader. If each image received from the camera can be mapped onto a cartesian plane, those below some calibrated offset on the Y axis can be considered to be currently passing through the RFID reader’s range, and those above that offset are considered the next set of people to enter the RFID reader’s range. The people entering the RFID reader’s range are counted, and their unique identifier is destroyed. The process in its entirety repeats on the next group of people in line.

The data is then sent to the database interaction subsystem to be compared against data from the RFID subsystem. The number of people currently passing through the RFID reader’s range is used for error or non-ticketholder entry.



**Figure 2.2c: Thermal Imaging software process flowchart**

References:

[6] “What Are the Different Types of RFID Technology?” *lowrysolutions.com,* Dec. 12, 2014. [Online]. Available:<https://lowrysolutions.com/blog/what-are-the-different-types-of-rfid-technology/> [Accessed Sept. 2, 2020].

[7] “What is a Bluetooth beacon?” *beaconstac.com*, [Online]. Available: <https://www.beaconstac.com/what-is-a-bluetooth-beacon> [Accessed Sept. 12, 2020].

[8] N. Chandler, “What’s an NFC tag?” *howstuffworks.com*, [Online]. Available: <https://electronics.howstuffworks.com/nfc-tag.htm> [Accessed Sept. 13, 2020].

[9] “Ultra-Wideband (UWB),” *bleesk.com*, [Online]. Available: <https://bleesk.com/uwb.html> [Accessed Sept. 13, 2020].

[10] FM-503. [Online]. *AliExpress*. Available: <https://www.aliexpress.com/item/4000133201826.html?spm=a2g0s.9042311.0.0.27424c4dpZPuIN> [Accessed 10 Sept. 2020].

[11] Sparkfun M6E Nano. [Online]. *Sparkfun*. Available: <https://www.sparkfun.com/products/14066> [Accessed 10 Sept. 2020].

[12] UHF-105. [Online]. *AliExpress*. Available: <https://www.aliexpress.com/item/32325789868.html?spm=2114.12057483.0.0.49ca74d4EMgOvq> [Accessed 12 Sept. 2020].

[13] FA-305A. [Online] *AliExpress*. Available: https://www.aliexpress.com/item/4000927334958.html [Accessed 14 Sept. 2020].

[14] Arduino Uno Rev3. [Online]. *Arduino*. Available: <https://store.arduino.cc/usa/arduino-uno-rev3#:~:text=Arduino%20Uno%20is%20a%20microcontroller,header%20and%20a%20reset%20button>. [Accessed 14 Sept. 2020].

[15] BeagleBone Black. [Online]. *BeagleBoard*. Available: http://beagleboard.org/black [Accessed 17 Sept. 2020].

[16] Raspberry Pi 4 B. [Online]. *RaspberryPi*. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/ [Accessed 14 Sept. 2020].

[17] F. Copes, “Arduino vs Raspberry Pi,” *flaviocopes.com*, [Online]. Available: https://flaviocopes.com/arduino-vs-raspberry-pi/#:~:text=The%20Raspberry%20Pi%20comes%20with%20a%2064%2Dbit%20microprocessor.&text=In%20terms%20of%20I%2FO,set%20of%20I%2FO%20pins.&text=Raspberry%20Pi%20runs%20an%20operating,Arduino%20is%20much%20more%20simple. [Accessed 15 Sept. 2020].

[18] PIM366. [Online]. *Digi-Key*. Available: https://www.digikey.com/product-detail/en/pimoroni-ltd/PIM366/1778-1232-ND/9606192 [Accessed 10 Sept. 2020].

[19] PIM365. [Online]. *Digi-Key*. Available: https://www.digikey.com/product-detail/en/pimoroni-ltd/PIM365/1778-1231-ND/9606191 [Accessed 10 Sept. 2020].

[20] FLIR ONE Pro. [Online]. *FLIR*. Available: https://www.flir.com/products/flir-one-pro [Accessed 10 Sept. 2020].

[21] “Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch and others,” altexsoft.com, Jun. 20, 2019. [Online].Available: https://www.altexsoft.com/blog/business/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/ [Accessed 15 Sept. 2020].