Lock Down

Team: 41

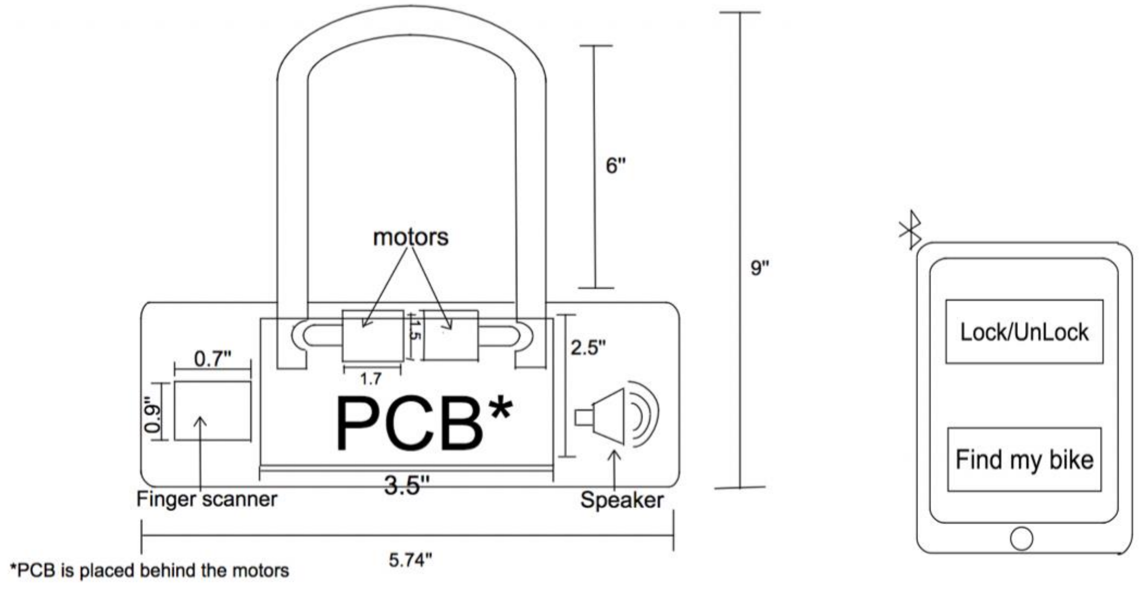
Section: 4

Locksmith

Tommy Agnello, Vaishnavi Dikshit, Ty Janik, Joseph Wang

Revision 4

Figure 1: Product Sketch



The lock is connected to the user's phone. To lock/unlock the bike, the user simply scans their finger that will trigger a locking circuit which will automate the process of locking/unlocking the user's bicycle. In case the scan is not working, the user can use our app on their phone. If the user is not able to find their bike, the user's phone can be used to locate the bike.

TA: Hengying Shan

Professor: Philip E. Pare

Contents

Revision Log

Revision	Date	Change #	Description
1	9/3/2020	1	Initial release.
2	9/10/2020	2	Activity Diagram updated
		3	Added system Block diagram
		4	Removed System Activity Description
		5	Added Subsystem Definitions
		6	Added additional requirement for bike security
3	10/15/20	7	Replaced the H-bridge with L293E in subsystem 3
		8	Updated Subsystem 3 block diagram
		9	Updated Subsystem 3 schematic
	10/17/20	10	Updated Subsystem 3 Specifications
		11	Updated Subsystem 4 Specifications
	10/22/20	12	Updated System Block Diagram
		13	Updated Subsystems 1 and 2
4	12/2/20	14	Updated Subsystems 1 and 2

		15	Updated System Block Diagram
		16	Updated Subsystem 4 schematic
		17	Added measurements for Subsystem 3 and 4
	12/3/20	18	Added Glossary, PCB Design, and Integration Plan

Table 1: Revision Log

Glossary

Advertising: Packets that are broadcasted by the peripheral when it is seeking a connection; these packets contain data to help the centrals identify the device's identity.

BLE: Bluetooth low energy, also known as Bluetooth v4.0

Bonding: A secure Bluetooth connection in which security parameters are stored as to not repeat the pairing process when the connection resumes.

Central: A Bluetooth capable device which initiates connections with peripherals and reads from and writes to the peripheral's characteristics.

FAR: False Acceptance Rate; Given as percentage of attempts a non-enrolled fingerprint will be detected as enrolled.

FRR: False Rejection Rate; Given as percentage of attempts an enrolled fingerprint will not be recognized.

GATT: Generic Attribute Profile; Describes how BLE data is transported and organized.

LDO Voltage Regulator: Low Drop-Out Voltage Regulator; DC to DC voltage converter that steps a higher voltage down to a lower one.

Motor Controller: a device or group of devices in an electrical circuit that can predetermine the performance of an electric motor

Peripheral: A Bluetooth capable device containing a GATT structure that the central interacts with.

Reverse Polarity Protection: Circuitry on a power input that diminishes any negative voltage to a safe level to prevent damage to the system. User to protect against user error of plugging power source in backwards.

Solenoid: A type of electromagnet that can generate a magnetic field through a coil wound into a tightly packed helix.

UUID: Universal Unique Identifier; A 128-bit number that devices use to reference members of another device’s GATT structure.

Executive description

The current bike lock designs take time and efficiency of people’s day when utilizing a bike as a mode of transportation. Another issue involving the current design is that the user cannot find their bike in case they forgot where they parked the bike. Our goal is to reduce the time necessary to lock/unlock one’s bicycle. Our solution includes using a fingerprint sensor to authenticate the owner of the bicycle and trigger a locking circuit which will automate the process of locking/unlocking the user’s bicycle. The lock will be connected to the user’s phone GPS to aid them in locating the bike. There have been phone controlled automatic bike locks such as the Linka lock[1] but they don’t consist of any GPS system.

User story

Bob the biker who frequently makes stops on his rides for coffee. He has gotten tired of the older, more tedious way of opening his bike lock that requires two hands. Bob is looking for a faster method of authentication that does not compromise security. Ideally, he would like to be able to open his lock with the touch of a fingerprint scanner that exists on the lock itself. Assuming the biometric occasionally has difficulty authenticating the user due to unexpected moisture, Bob does not want to experience any undue inconvenience using his lock. For this reason, he would like the lock to send his phone a notification over Bluetooth so he can quickly verify his identity.

Billy is another biker who lives downtown and frequently changes his routes to keep his rides more interesting. With thousands of bike racks in his city and a lack of consistency with his bike placement, Billy sometimes forgets exactly where he last locked up his bike. He would like the ability to track where his bike is when it is locked so he can get a good idea of which rack he used.

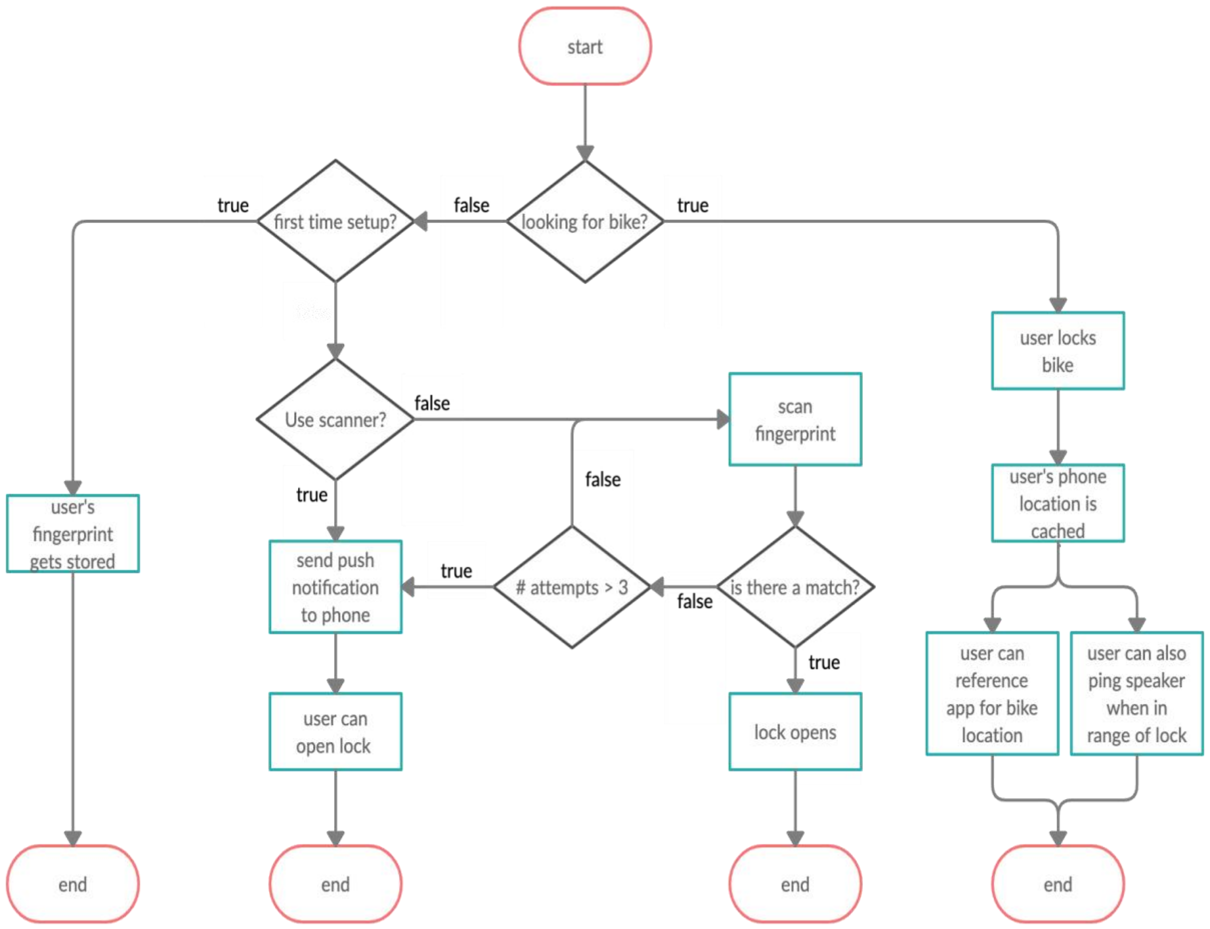
Requirements

1. Must unlock from user’s fingerprint
2. Must unlock from user’s phone using our app if the fingerprint scanner fails
3. Must reject non-user phones using our app
4. Must reject non-user fingerprint
5. Must use the phone's GPS system in case the user forgets where the bike is located.
6. The app must be able to store lock’s GPS.
7. The GPS system of the user’s phone must be able to connect to the bike lock in a bluetooth range of 15 feet
8. Must lock from only the user’s fingerprint.
9. Must lock from user’s phone using our app if the fingerprint scanner fails
10. App must connect to the speaker via bluetooth.

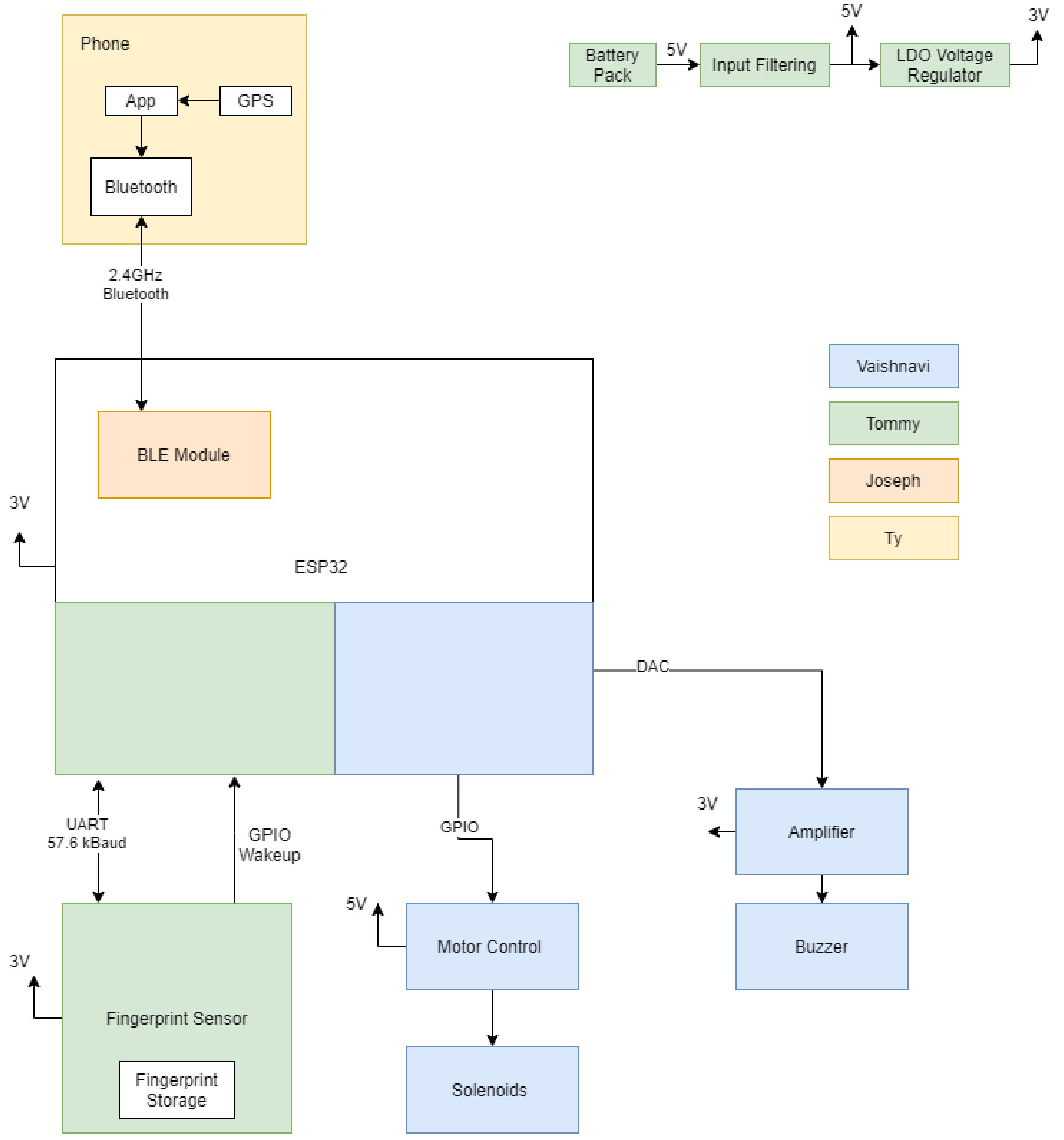
Factors influencing requirements

1. The requirement to include the phone's GPS system will help the user find the bike easier in case the user forgets about the location.
2. The system can be used in many places throughout the world, but focus on English speaking regions for simplicity
3. No cultural factors were considered while designing since the lock system can be used by anyone with a phone device.
4. Allows the user to not worry about carrying an extra key when out
5. Environmental factors are a part of design consideration to minimize impact, but are not among the requirements. This device trades convenience for increase in energy use and e-waste.
6. Economic factors that influence our system include the cost. We plan on maintaining the cost less than what the current automatic bike locks cost.

System Activity Diagram



System block diagram



Subsystem 1: Fingerprint Reader

Subsystem Owner: Agnello

Specifications

Requirement	**Specification**	**Justification**
Must allow an authorized user to lock/unlock using fingerprint scanner	The system should send a accept message when an authorized fingerprint is found	Must allow the user to open the lock.
	The system should send a reject message when an unauthorized fingerprint is found	Must be able to keep out unauthorized users
	Idle current draw of 2uA	Allows the battery to last longer, thus allowing requirements to be met for longer.
	False Acceptance Rate < 0.001% and False Rejection Rate < 1%	Must determine whether it should accept with certain accuracy

Table 2: Specification of Fingerprint Subsystem

Sub-system diagrams

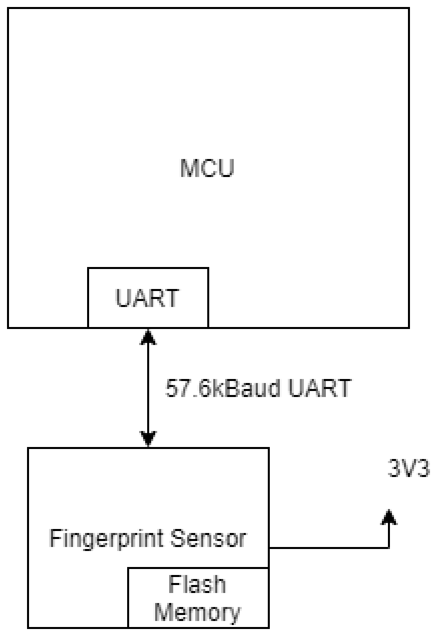


Figure: Fingerprint Sensor Block Diagram

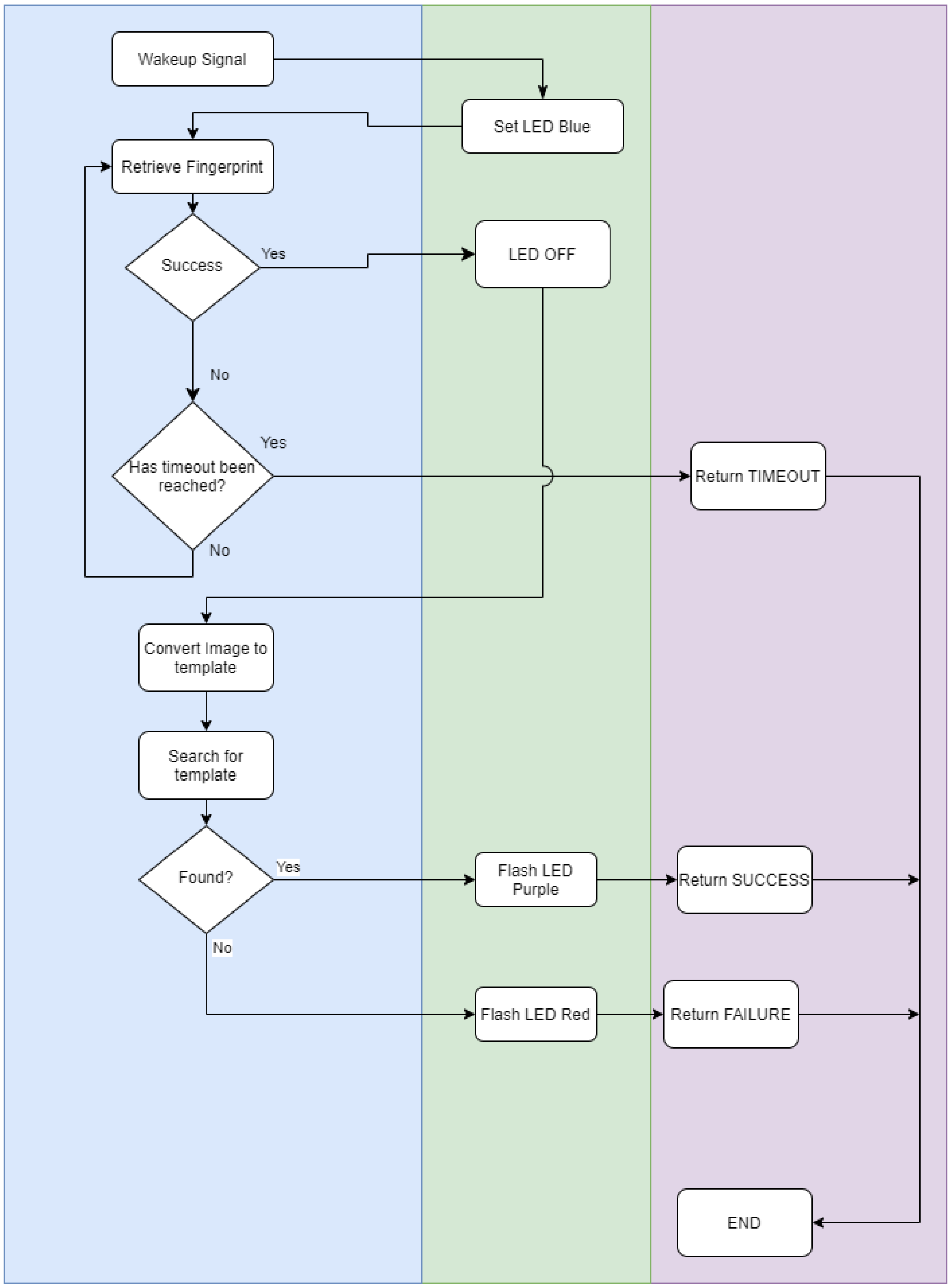


Figure: Fingerprint Verify Flow Diagram

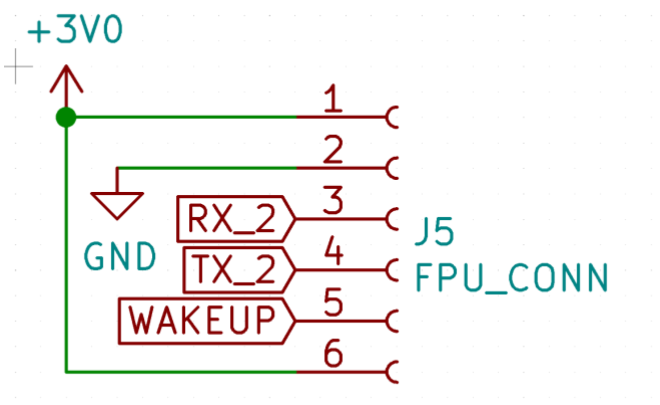


Figure: Fingerprint connector schematic

Sub-system interactions

The sensor communicates with MCU over 57.6kBaud UART.

Core ECE design tasks

ECE 362 - Microprocessors provided a lot of knowledge needed for this subsystem. It goes over how to integrate modules with microcontrollers and to search through documentation to find the needed information.

Schematics / parts / algorithm

The 100018754 BM-Lite was the intended fingerprint reader. After complications, the decision has been made to switch to the Adafruit R503 for easier integration. The documentation is much richer for this line, which should help move the project along.
The manufacturer has a library with useful functions to handle the low level operation of the device. I took that library, and extended its functionality for ease of integration. The end result is a class that inherits the class from the library. New methods were created, such as finger.enroll() and finger.verify(), which allow for one line calls. That is found at https://github.com/Roosta078/custom_finger, along with a demo sketch to highlight its functionality.

Theory of Operation

Fingerprint Reader

Fingerprint scanning is a popular authentication method; due to its speed, low False Acceptance Rate(FAR) and lack of a password to remember. It works by first getting a raw image of the user's fingerprint. It takes this raw data and maps key features to the data. These key features are then mapped to a template, and stored. Authorization occurs when a sufficient number of key features are matched on both the stored template, and newly scanned fingerprint. This can be done quite easily on specialized HW, which is why most sensors handle all storage/processing onboard and just communicate results over a data bus.

Specification measurements

The first two specifications were met, evidenced by the ability for the lock to unlock from the user's fingerprint, and not from a different finger. The idle current draw was measured to be 2.6uA, which is slightly above spec, but will have a negligible impact. The exact accuracy of the sensor was not able to be obtained, but it subjectively performed on par with cell phone fingerprint sensors.

Standards

UART Standard is being followed. Separate RX and TX lines, with clock tolerance +-1.5%.

Subsystem 2: Power

Subsystem Owner: Agnello

Specifications

Requirement	Specification	Justification
Must allow an authorized user to lock/unlock using fingerprint scanner	The system should filter the battery input before seeing any other circuitry	There must be a mechanism to prevent damage to device from incorrect battery insertion. Despite feedback saying this is unnecessary, I still believe it to be extremely naive to not include, thus it is included.
	The system should output approx 3V, for IC operation	Must be able to power chips
	The system should output approx 5V, for motor operation when needed	Must be able to drive solenoids

Table 3: Specification of Power Subsystem

Sub-system diagrams

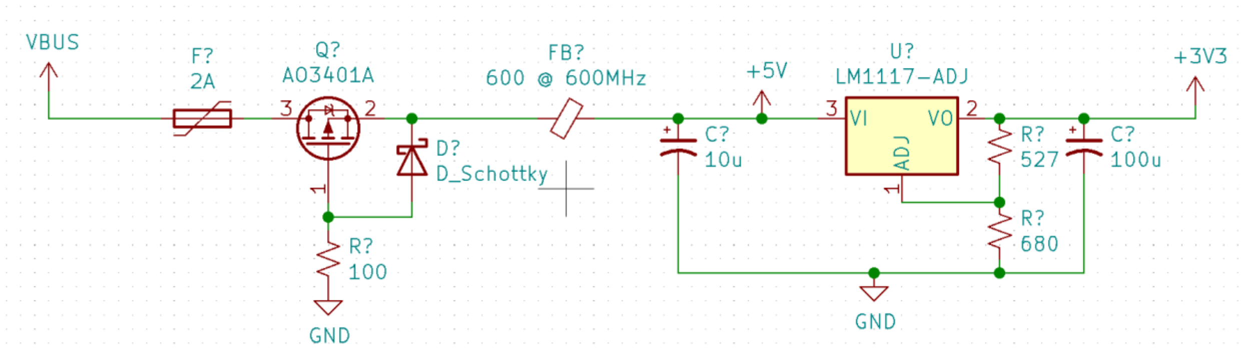


Figure: Power Subsystem Schematic

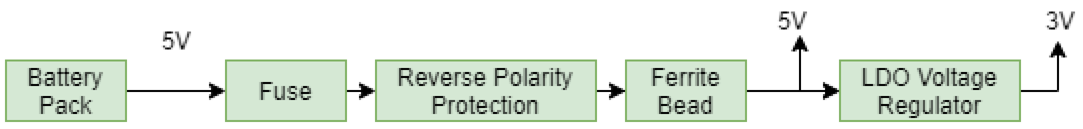


Figure: Power Subsystem Block Diagram

Sub-system interactions

The system is responsible for supplying clean power to the rest of the system at 5 or 3V. It will ensure that the delivered 5V is clean and within current limit, and that the 3V is stable.

Core ECE design tasks

LDO's were not covered in my ECE courses. Buck converters were discussed, but not used for this project due to the increased complexity and that their feature set was not necessary.

Schematics / parts / algorithm

The LM1117-ADJ was chosen because it met our project needs and the ECE shop had it on hand. It is able to supply the necessary voltage with more than enough current. The output is shown to be stable over a range of inputs.

The USB input is fused to protect against shorts and overvoltage. The PMOS and Schottkey diodes provide reverse polarity protection against negative voltage. The ferrite bead helps to reduce any noise in the input voltage.

Theory of Operation

Boost Converter

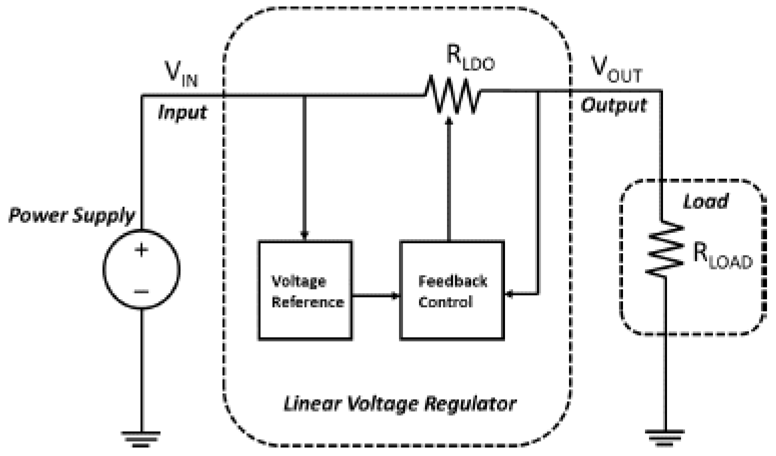
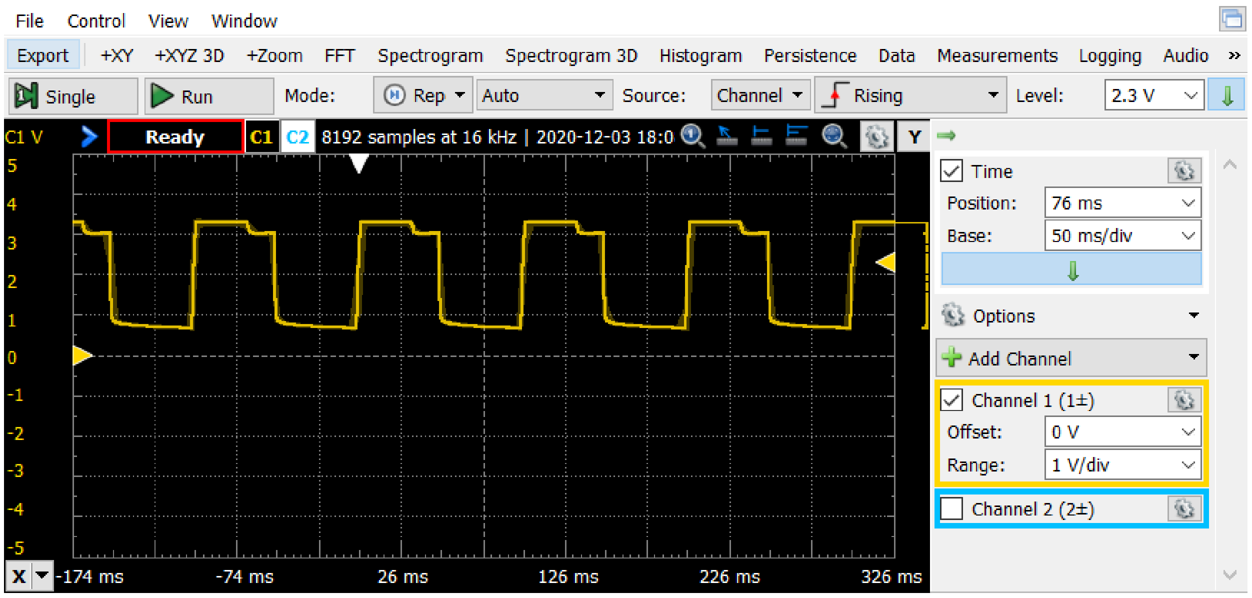


Figure: Simplified Linear Voltage Regulator

The LDO voltage regulator works by acting as a voltage controlled resistor. It will change its impedance according to the output voltage. It can increase the output by decreasing its impedance, as there will be a voltage drop across the regulator. This leads to a lower efficiency, compared to switching voltage converters. This was deemed acceptable for our project, as we have a very low current draw on that rail.

Specification measurements



Mode: Rep Auto Source: Channel Rising Level: 2.3 V

8192 samples at 16 kHz | 2020-12-03 18:0

Time
Position: 76 ms
Base: 50 ms/div

Channel 1 (1±)
Offset: 0 V
Range: 1 V/div

Channel 2 (2±)

Figure: Demo of Reverse Polarity Protection

The reverse polarity protection is shown above to work effectively. A square wave alternating from +3.3V to -3.3V was inputted to the system. The input filtering was able to squash the negative voltage input, in order to protect the other components.

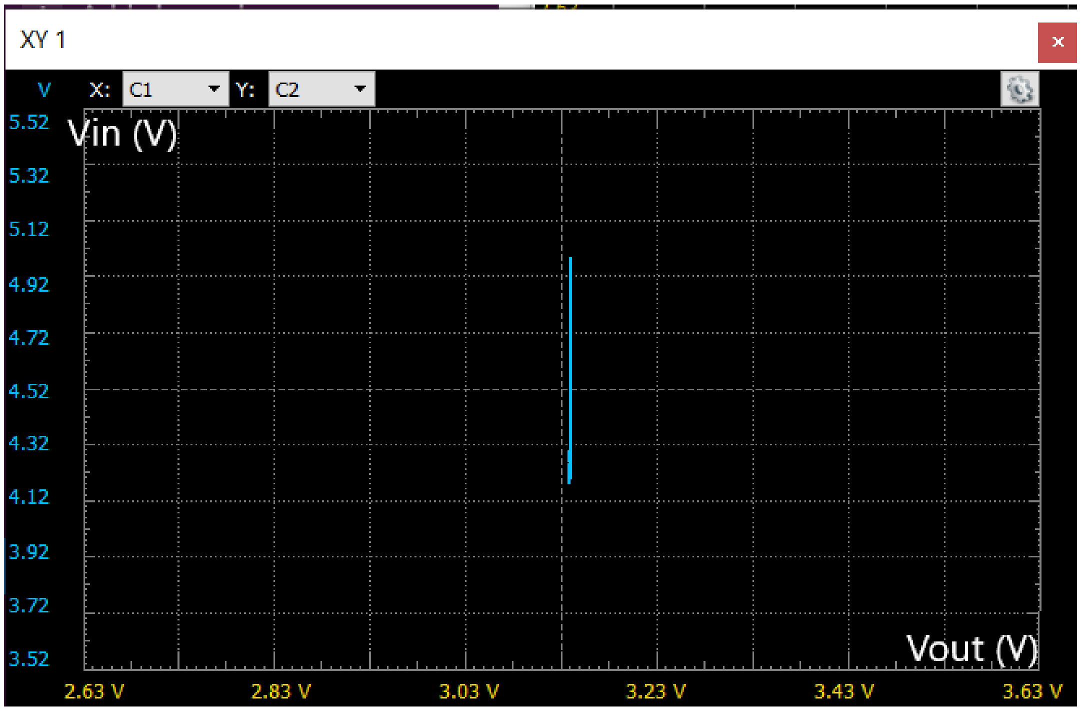


Figure: Demo of LDO Voltage Regulator

The voltage regulator is shown to output a stable 3V output for a range of 5V inputs. Due to limitations of the AD2, I was unable to test voltages above 5V, but it is believed to perform similarly. The system is also able to pass-through the 5V input from the USB input.

Standards

N/A

Subsystem 3: Solenoid Control

Subsystem Owner: Vaishnavi Dikshit

Specifications

Requirement	**Specification**	**Justification**
Must allow an authorized user to lock/unlock	The motor controller should be able to run 2 solenoids	There should be 2 solenoids to lock the shackle in place. This will secure the lock on the bike. The solenoids will

using fingerprint scanner		fit into holes which will be on the sides of the U part to lock the bike and the solenoids will be released from those holes to unlock.
	The motor should have an output current less than 1.2A	The solenoids need at least 200mA to work according to the datasheet
Must allow an authorized user to lock/unlock using mobile app	The solenoids should be able to operate between the range of 3.3V to 5V	The power subsystem provides 5V for the motor controller and hence the solenoids should work in that voltage range to function properly
The subsystem should only be powered when the fingerprint or app is in use	The motor controller should be able to control Latching solenoid only for 2 seconds	To avoid overheating of the systems as solenoids tend to produce heat if they are continuously running. This will also help in power consumption as locks are only used to lock and unlock. Once the solenoid is in the desired position there is no need to power them.
The system must secure the bike	The solenoids should have a stroke of at least 5mm and force of 1.5 N	The solenoid should have a stroke long enough to fit in the holes of the lock and secure it
The system should be lightweight	The solenoids should weight less than 0.17 lbs	The lock should be portable and hence should not be heavy.

Table 4: Specification of Solenoid Subsystem

Sub-system diagrams

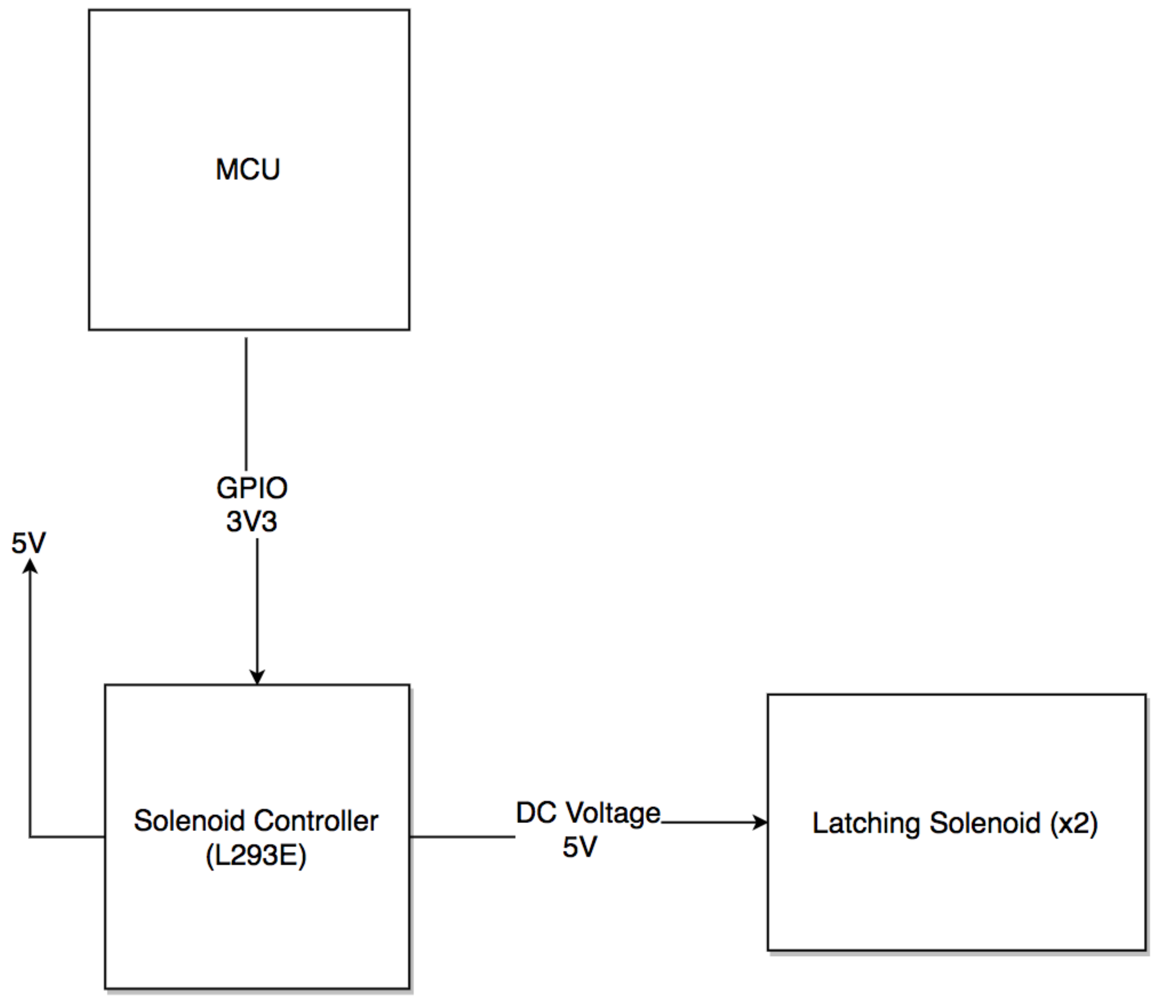


Figure: Subsystem Block Diagram

Sub-system interactions

1. Power Subsystem: The power subsystem will drive the motor controller and the solenoids when the app or fingerprint scanner is used. The 5V is needed for only a short period of time. When the lock is in locked or unlocked state no power should be used by the subsystem. The enable pin in the solenoid controller chip will be used to monitor the power supplied to the system.

2. Microcontroller/ Bluetooth: The mobile app should be able to control the motor controller via bluetooth connection.

3. Fingerprint subsystem: The fingerprint should be able to control the subsystem when the owner's fingerprint matches.

Core ECE design tasks

ECE 321: Electromechanical Motion Devices will provide most knowledge to implement the motor control subsystem as it goes over the different types of motor used and how they work.

ECE 323: Electromechanical Motion Devices and Systems Laboratory will help with the implementation of the motor and the hall effect sensor. And how to use them in measuring the position.

Schematics / parts / algorithm

L293E is used as the motor driver to control the push/pull direction of the solenoids. It utilizes H-bridge circuit which helps in bidirectional movement of the solenoid. The IC can run 2 motors/solenoids. Latching solenoids will be used for the locking and unlocking.

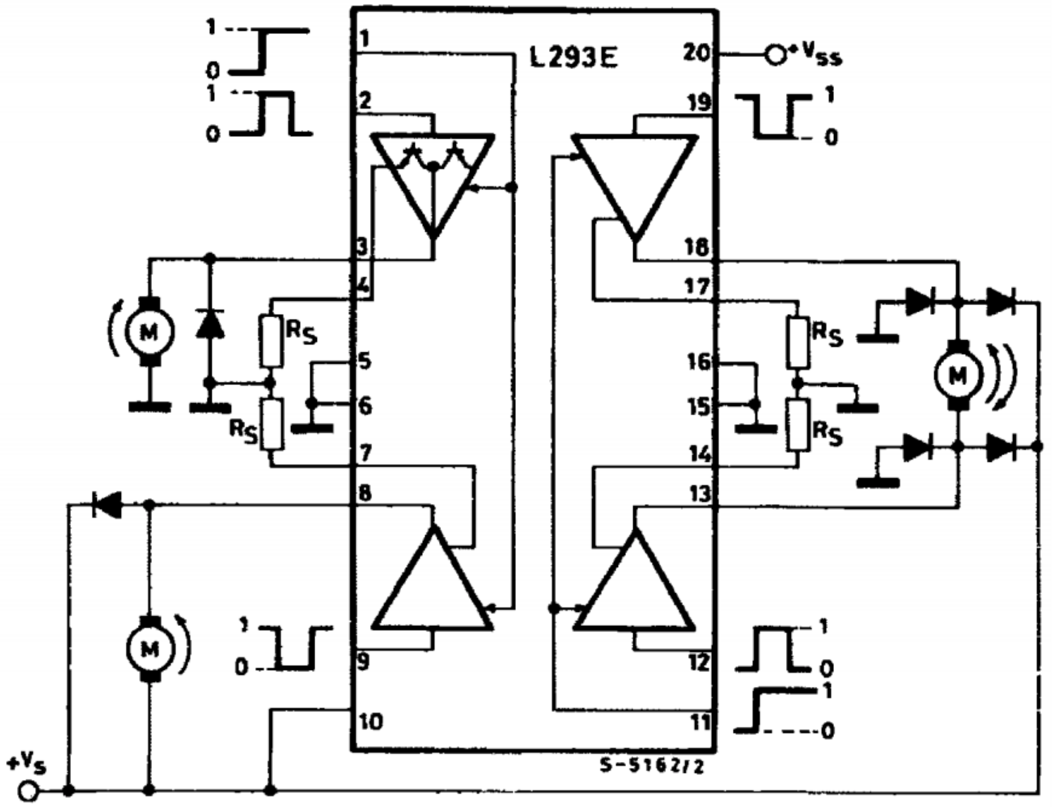


Figure: IC schematic from [2] L293E data sheet

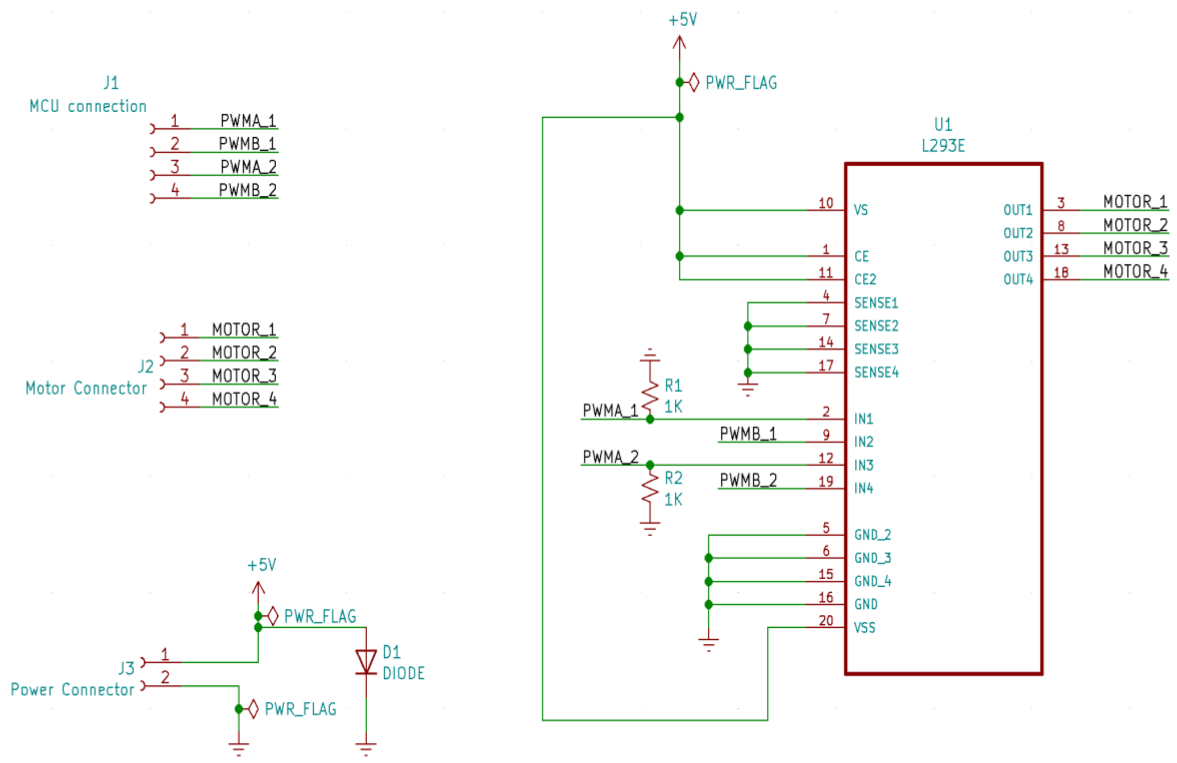


Figure: Schematic of the subsystem

Note: The schematic is based on schematic for L293E driver datasheet

Pins 4,7,14, and 17 are connected to ground as in this application there is no need for sense resistors and since there is no need they are connected to ground.

All the enable pins and source pins are connected to 5V power supply from the power subsystem.

Pull down resistors are used at one of the inputs for each side to provide the correct polarity to the solenoids.

Removed sensors because that was adding extra hardware to the subsystem. The sensors can be substituted by code. Since there are only 2 directions that the solenoids can move in, using MCU to control that can be much easier than using sensors. If later in the process the need for external sensors is required then the esp32 on chip hall effect sensors can be used.

The output of the system is connected to the 2 solenoids and input is connected to the GPIO pins of the ESP32. The GPIO pins will provide 3.3V when high and 0V when low.

Using the bluetooth module of the microcontroller, the app can control the locking mechanism as well.

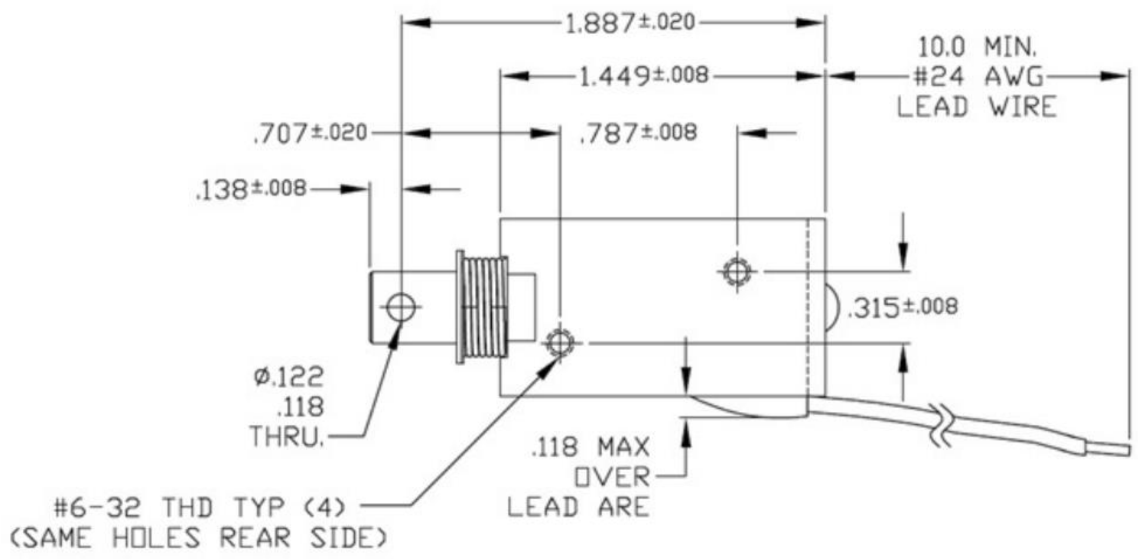


Figure: Solenoid specifications

Parts:
Linear Solenoids (x2) - B14-L-258-B-4
Motor Driver (H-bridge) - L293E
(x2) 1K pull down resistors

Theory of Operations
<u>Solenoid Driver</u>
The solenoid driver used is L293E. The driver is a push pull four channel driver, which is capable of delivering output currents to 1A per channel. The H-bridge in the driver uses transistors and hence are not desirable if the solenoids are running for long time periods. In this case the driver is inexpensive and ideal as they only control the solenoids for a few seconds. Each channel is controlled by a TTL compatible logic input and each pair of drivers (a full bridge) is equipped with an inhibit input which turns off

all four transistors. A separate supply input is provided for the logic so that it may be run off a lower voltage to reduce dissipation. It will communicate to the ESP32 using GPIO pins and communicate to the mobile app using the bluetooth module of ESP32. The solenoid driver utilizes 3.3V GPIO logic from the microcontroller to move the solenoid in the desired direction. When the fingerprint sensor detects the owners' fingerprint or the owner uses the app to control the bike lock, the subsystem is activated. If the solenoids are in pull state (unlocked) then input 1 and 3 is high and input 2 and 4 are low to push the solenoid (locked). And when the solenoids are in pull state (locked) then input 1 and 3 is low and input 2 and 4 are high to push the solenoid (unlocked).
R1 and R2 are used to provide the right polarity for the solenoids. One of the inputs is pulled to low so that reverse polarity can be achieved. 1K resistors are used for this.

Sense 1,2, 3, 4 are connected to ground.

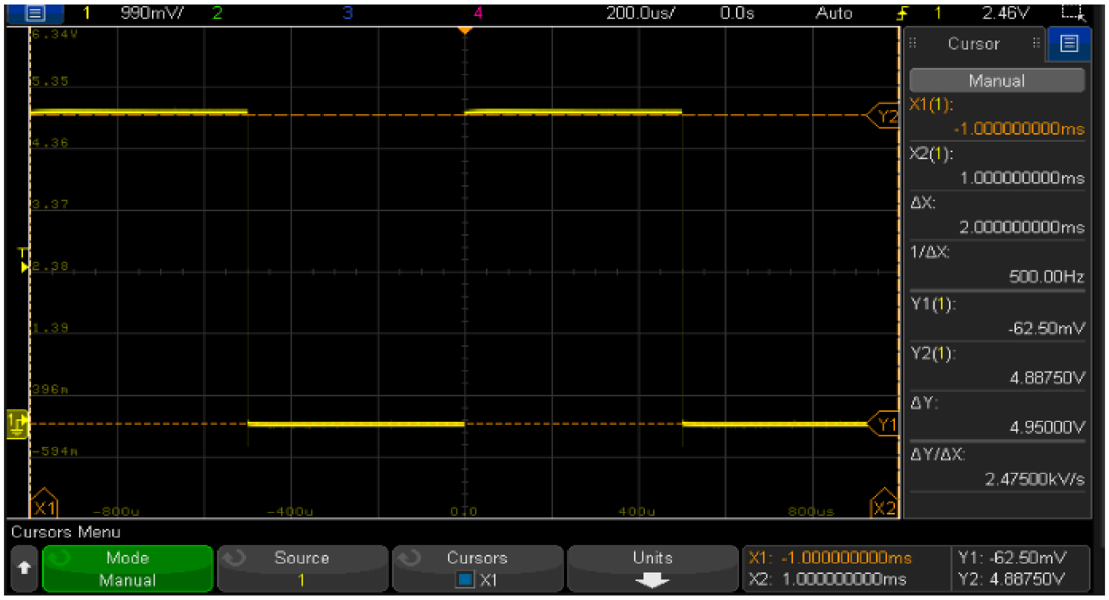
ENBL 1&2: These pins should be high and between 2.3V to 5V to enable the pin. When LOW (0V) the driver becomes disabled.

This design is inexpensive and low power and compatible with the microcontroller GPIOs.

<u>Solenoid</u>
Latching solenoids are implemented in this subsystem. Latching solenoids are well suited to applications where the moving time is very short compared to holding time in the closed position. Which is the case with this project. Latching Solenoids are extremely well suited to locking mechanisms. They can be driven in either direction and can hold in either end position with no power applied. A short power pulse is all that's needed to stimulate movement. The solenoids need 200mA power to function and hence compatible with the driver which can deliver up to 1A at each output.

Specification Measurements
The main purpose of the subsystem was to provide a locking mechanism which was provided using the H bridge (L293E). The chip provides opposite polarity when different inputs are passed. Depending on inputs from the app or the fingerprint scanner, the inputs to L293E were changed. Here is the measurement for different state polarities. The first figure shows the output of pin 3. The solenoids are in pull state and

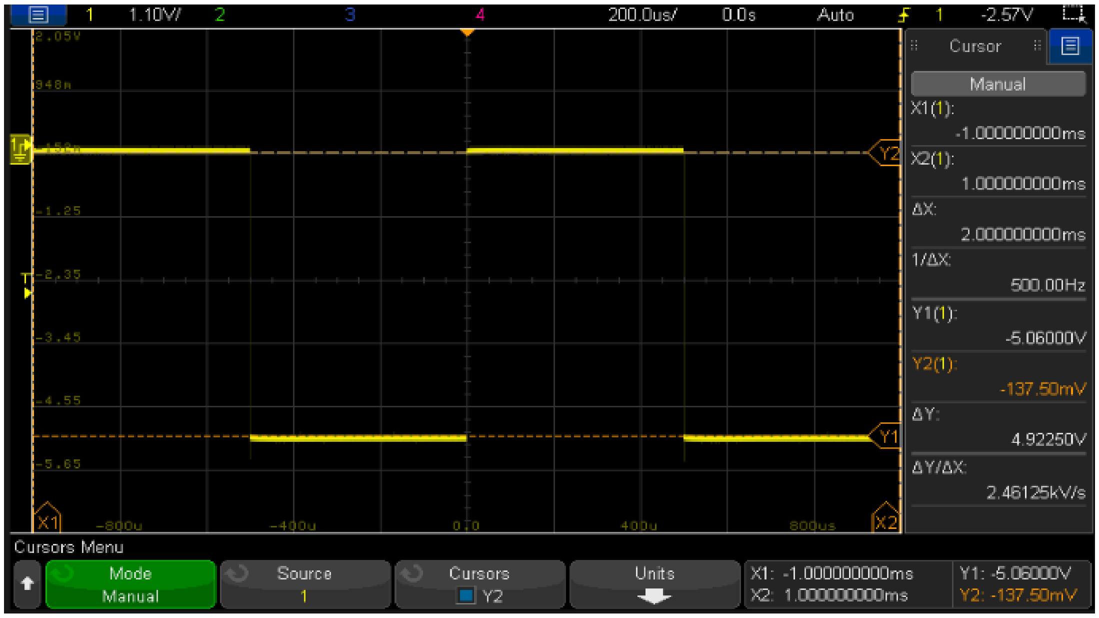
getting an output of a square wave of 0V to 5V. Next figure is the scope plot of the unlocked state. This is when the solenoids are in pull state and show reverse polarity. The figure is output from pin 13.



```
Cursor
Manual
X1(1):  -1.000000000ms
X2(1):  1.000000000ms
ΔX:     2.000000000ms
1/ΔX:   500.00Hz
Y1(1):  -62.50mV
Y2(1):  4.88750V
ΔY:     4.95000V
ΔY/ΔX:  2.47500kV/s

X1: -1.000000000ms   Y1: -62.50mV
X2: 1.000000000ms    Y2: 4.88750V
```

Figure: Locked state (Push state of solenoid)



```
Cursor
Manual
X1(1):  -1.000000000ms
X2(1):  1.000000000ms
ΔX:     2.000000000ms
1/ΔX:   500.00Hz
Y1(1):  -5.06000V
Y2(1):  -137.50mV
ΔY:     4.92250V
ΔY/ΔX:  2.46125kV/s

X1: -1.000000000ms   Y1: -5.06000V
X2: 1.000000000ms    Y2: -137.50mV
```

Figure: Unlocked state (Pull state of solenoid)

Current consumption of the lock is around 1.15A which is less than the peak current that the power system can provide. The Power consumption when the solenoids are enabled is 7W. When they are idle there is no current supply to the solenoids but Vss and Vs are still connected to 5V and hence there is some current consumption. Since it is close to zero it can be neglected.

Standards

RoHS standards for motors are being followed.

Osha heat standard for solenoid is being followed

Subsystem 4: Speaker

Subsystem Owner: Vaishnavi Dikshit

Specifications

Requirement	**Specification**	**Justification**
Must connect to app via bluetooth	The speaker should be compatible with ESP32 bluetooth module	To connect to the mobile app, the speaker should be compatible with the bluetooth module of ESP32.
Must be loud enough so that the user can hear the speaker	The amplifier should output 110mA a	The buzzer draws more current than the ESP32 I/O pins can supple and hence a common emitter amplifier is used
Must be portable and lightweight	The buzzer should weigh less than 0.17 lbs and less than 3 diameter in size.	The buzzer should be light so it's convenient for the user to carry the lock around.

Table 5: Specification of Speaker Subsystem

Sub-system diagrams

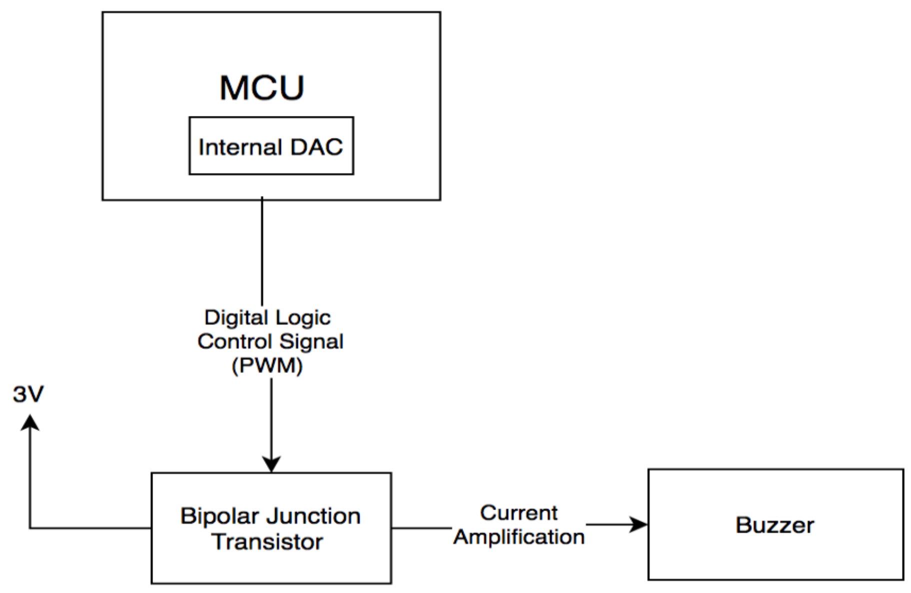

MCU
Internal DAC
Digital Logic Control Signal (PWM)
3V
Bipolar Junction Transistor
Current Amplification
Buzzer

Figure: Speaker subsystem

Sub-system interactions

1. Power Subsystem: The power subsystem will drive the speaker with 3V.

2. Bluetooth module/ESP32: The speaker will be connected to the mobile app via bluetooth module of ESP32.

3. Mobile App: The mobile app button 'find my bike' will communicate with the speaker. When the app and ESP32 are within the bluetooth proximity. They are connected and then the app can be used to activate the speaker to find the bike.

Core ECE design tasks

ECE 207: Electronic measurement techniques will help with the implementation of the speaker and ECE 255: Introduction to Electronics Analysis and Design will help in understanding how BJT can work as an amplifier and how the 3 terminals can be used to operate the speaker.

Schematics / parts / algorithm

2N2222N is an NPN Bipolar Transistor that will be used as an amplifier. The Transistor's base terminal is biased in a way as to forward bias the base-emitter junction to act as an amplifier. A buzzer with max current draw of 110mAand voltage supply of 3V to 6V is used and connected to the collector terminal of the BJT. The schematics below explains the connection

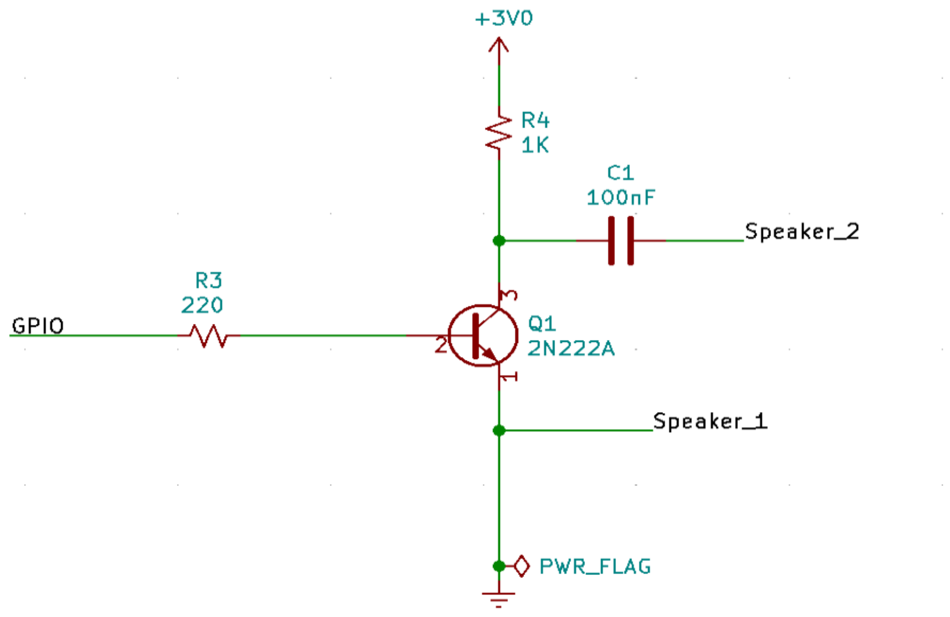


Figure: Speaker subsystem schematics

Parts
2N2222 transistor
Speaker - 3" Diameter - 4 Ohm 3 Watt
https://www.adafruit.com/product/1314?gclid=Cj0KCQjwuL_8BRCXARIsAGiC51D5OvpgaeZP0oAcPRGMAhCpW_WhOis9Au9mPPZmR5v2pGtdcqpIIOMaAriTEALw_wcB
220 ohm resistor

Theory of Operation
A 2N2222 NPN transistor is being used to connect the speaker to the Microcontroller. Connecting the speaker directly will result in low volume and noise. The reason for softness is that the digital out pins can only source or sink around 40mA. The transistor with 220 ohm resistor at the base of the transistor is used to amplify the current at the pin out of the microcontroller. The circuit is a class A amplifier, which is biased around the Q-point within the middle of it's load line shown in the figure below. Since it is biased in the middle, it is never driven into the cut-off or saturation region. It stays in a linear region and conducts current over the full 360 degrees of input cycle. This was the best choice due to good linearity, high gain, and low signal distortion. One main disadvantage is the power consumption since it's always in a linear region. But in this

case, our lock only needs it for a few minutes and can be controlled through the app and hence it will be on for a small period of time. Using the GATT of the microcontroller, the spreaker will be connected to the app.

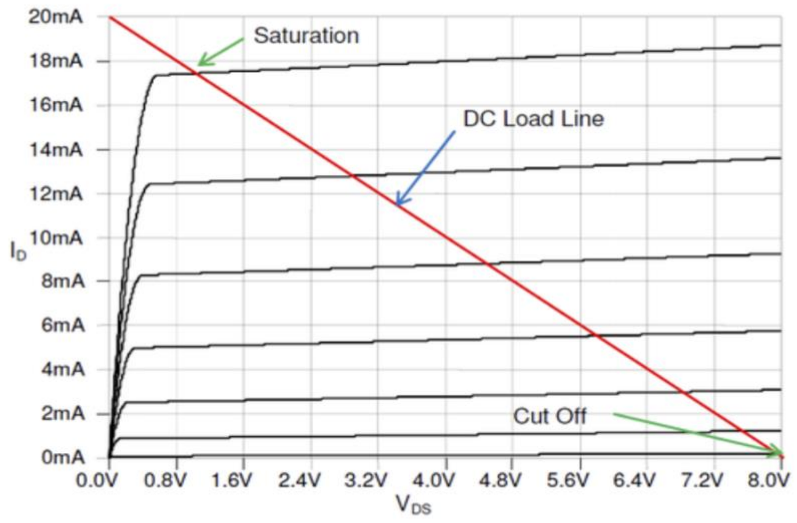
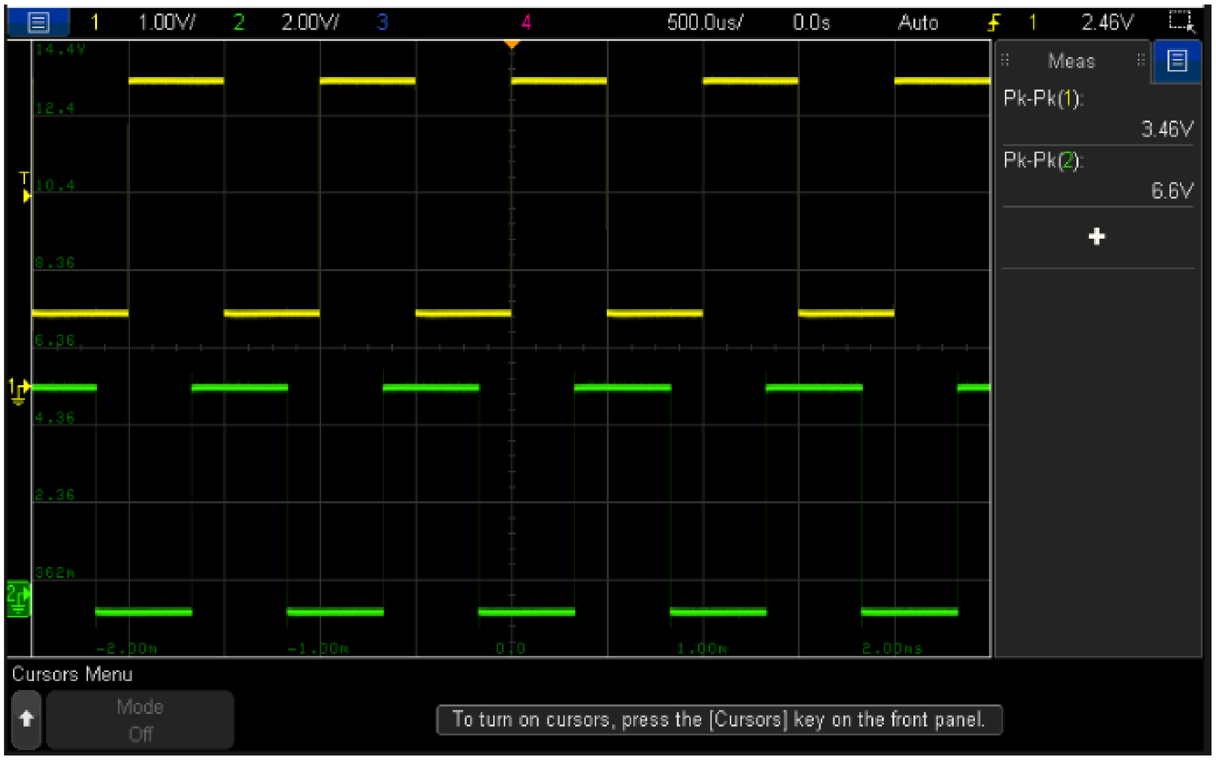


Figure: IV Characteristic Plot for NPN Transistor

Specification Measurements

In the table below, we can see that the output current generated by GPIO pins is 40mA which is not enough for the buzzer to operate. The output of the amplifier is 103mA which is close to the max current the buzzer works at and hence the amplifier was successful in amplifying the current. Other than the current the amplifier also amplified voltage which can be seen in the figure below.The yellow signal is input and green is output. The voltage gain was about 1.9 and current gain was about 2.83. The output is also inverted as the common emitter amplifier inverts the signal and hence it proves that the amplifier is working properly.

Input Current	40mA
Output Current	103mA



To turn on cursors, press the [Cursors] key on the front panel.

Figure: Input and output of the BJT amplifier

Standards

SIP10 resistance

FCC part 15

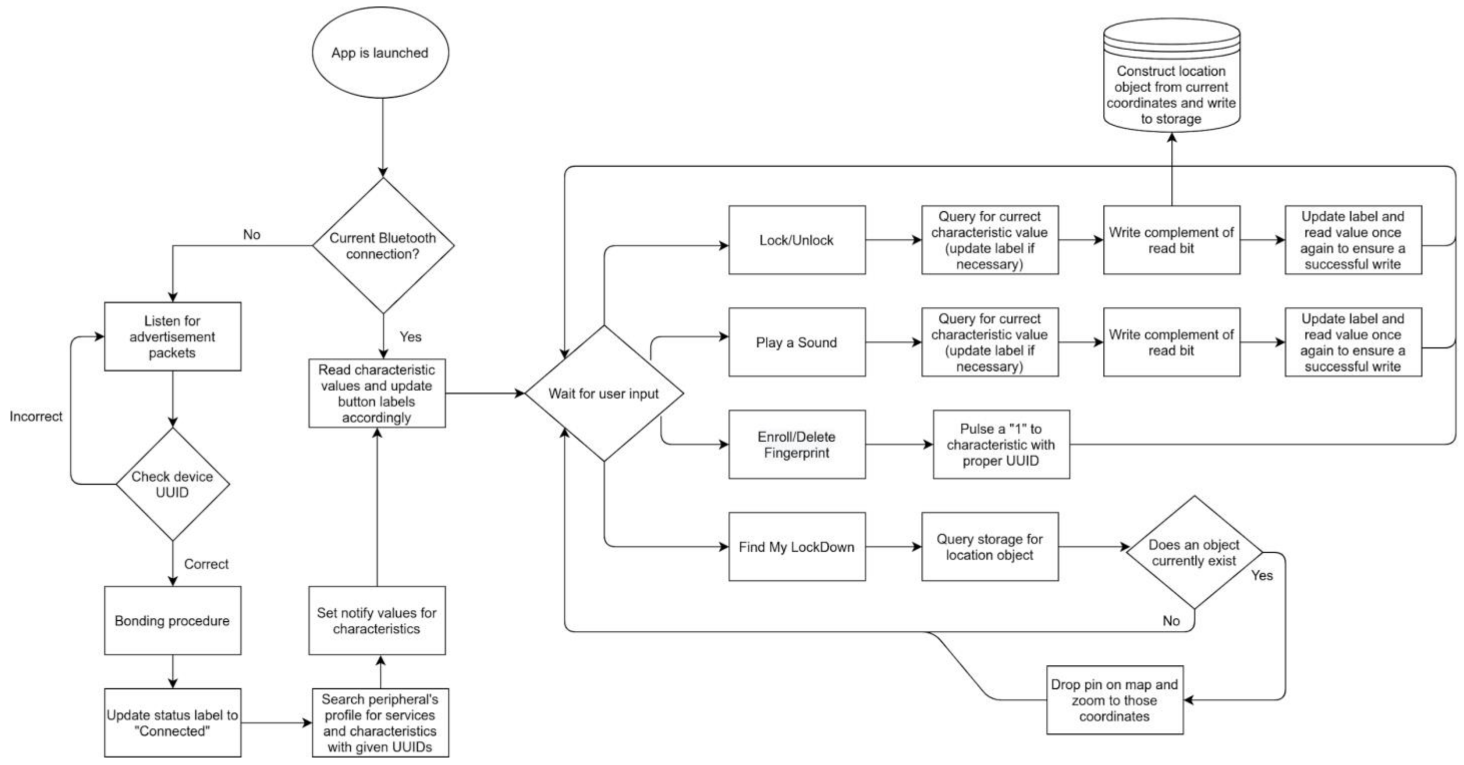
IEC 61672-2:2013

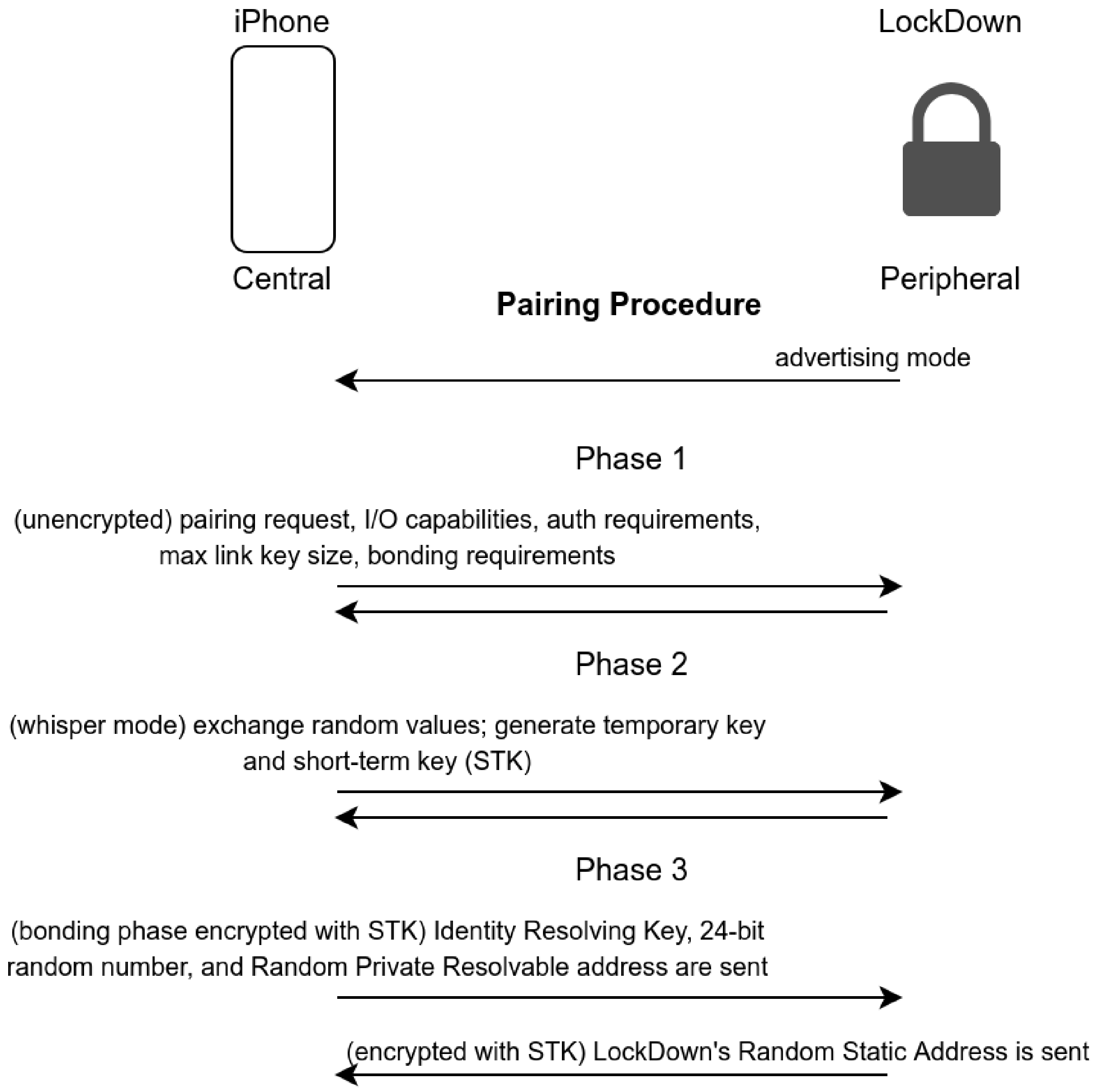
Subsystem 5: Mobile Application

Subsystem Owner: Ty Janik

Subsystem Diagrams

Software flow diagram:


App is launched
Current Bluetooth connection?
No
Yes
Listen for advertisement packets
Incorrect
Check device UUID
Correct
Bonding procedure
Update status label to "Connected"
Search peripheral's profile for services and characteristics with given UUIDs
Set notify values for characteristics
Read characteristic values and update button labels accordingly
Wait for user input
Lock/Unlock
Query for currect characteristic value (update label if necessary)
Write complement of read bit
Update label and read value once again to ensure a successful write
Construct location object from current coordinates and write to storage
Play a Sound
Query for currect characteristic value (update label if necessary)
Write complement of read bit
Update label and read value once again to ensure a successful write
Enroll/Delete Fingerprint
Pulse a "1" to characteristic with proper UUID
Find My LockDown
Query storage for location object
Does an object currently exist
Yes
No
Drop pin on map and zoom to those coordinates


iPhone
LockDown
Central
Peripheral
Pairing Procedure
advertising mode
Phase 1
(unencrypted) pairing request, I/O capabilities, auth requirements, max link key size, bonding requirements
Phase 2
(whisper mode) exchange random values; generate temporary key and short-term key (STK)
Phase 3
(bonding phase encrypted with STK) Identity Resolving Key, 24-bit random number, and Random Private Resolvable address are sent
(encrypted with STK) LockDown's Random Static Address is sent

Subsystem interactions

The mobile application will communicate via bluetooth protocol with the MCU's onboard bluetooth module. It will also use an API call to request the current location of the user's iPhone as long as the user has allowed access for this action.

Theory of Operation

The initial pairing/bonding process begins with the user powering on their lock. Once the lock has been turned on, it will enter advertising mode so that the mobile app can identify the correct peripheral device. After the device has been located, the two devices pair and the bonding requirements are stored so that this process will not need to be repeated.

For a user who has already connected to their device, the application will begin by reading from the peripheral's characteristics to update the button-text on the UI. Following this task, the app enters an idle state until a button is pressed. As seen in the flowchart above, pressing a button causes a series of events to be executed that will carry out the specified task, finishing with a return to the idle state as it waits for more user input. For example, if a user would like to know the last location at which they locked their bike, the app first checks to see if there are any location objects stored in nonvolatile storage. If there is not, the request is ignored. If one exists, the object is retrieved and a pin is dropped on the map using the latitude and longitude stored inside the object before returning to the idle state. For a task that references a GATT characteristic stored on the BLE module, the process is generally as follows: the characteristic is read to ensure the value has not changed since that last time it was referenced, the complement of the read value is written to the characteristic, and it is read once again to ensure the write was successful.

Specifications

Requirement	**Specification**	**Justification**

Must unlock from user's phone using our app if the fingerprint scanner fails	The Bluetooth connection must remain usable at a distance of about 5 meters. While within the 5 meter radius, the user must be able to write to the BLE server's "lockState" characteristic.	This gives the user a second method of unlocking their lock in case there are any issues with the fingerprint scanner such as moisture between the user's finger and the sensor
Must use the phone's GPS system in case the user forgets where the bike is located	The coordinates of the lock's last location are stored within the app and are accurate to at least ten meters	Cities tend to have a substantial amount of bike locks densely packed into one area, so such a feature gives the user the opportunity to be reminded of where they left their bike
Must lock from user's phone using our app if the fingerprint scanner fails	The Bluetooth connection must remain usable at a distance of about 5 meters. While within the 5 meter radius, the user must be able to write to the BLE server's "lockState" characteristic.	If the user has started to walk away from their lock and does not remember if they locked their bike, they can use their app to verify the state of their lock and even press the lock button if it was unlocked
The app must be able to connect to speaker using bluetooth	While the application is able to sustain a Bluetooth connection, it must be able to broadcast a signal that can be received and played by the onboard speaker to aid the user in finding their locked bike	This feature is useful for quickly locating the lock once the user is close to the rack at which they last left their bike to make up for some of the lack of precision in GPS

Core ECE design tasks

Developing a successful application requires knowledge from ECE 368 (*Data Structures*) to ensure that the data is processed efficiently. Writing code in a high-level language such as Swift, which is very similar to Python, utilizes knowledge from ECE 20875

(*Python for Data Science*). Finally, knowledge from ECE 404 (*Introduction to Computer and Network Security*) aids in ensuring the application's data is handled and transmitted securely.

Schematics / parts / algorithm

The app is written in Swift and utilizes Core Bluetooth to send data to the device as well as Core Location's tracking features. No parts are necessary for this subsystem. An iPhone X was used for testing because Xcode's built-in simulator lacks support for Bluetooth connections. For proper validation, an ESP32 was used to ensure data transmission operated as expected.

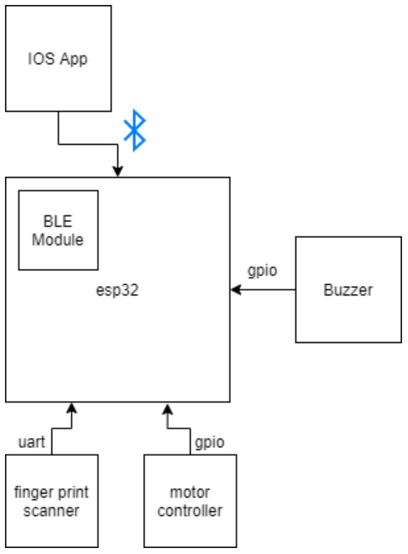
Standards

The transmission of data back and forth from the MCU to the mobile device will use the Bluetooth Low Energy protocol, implemented with CoreBluetooth. The application will use the database schema provided by Apple's Core Data for the utilization of persistent storage. Location information will be provided through the Core Location library with the help of MapKit to render the saved location onto a map for the user's convenience.

Subsystem 6 Bluetooth Low Energy Module

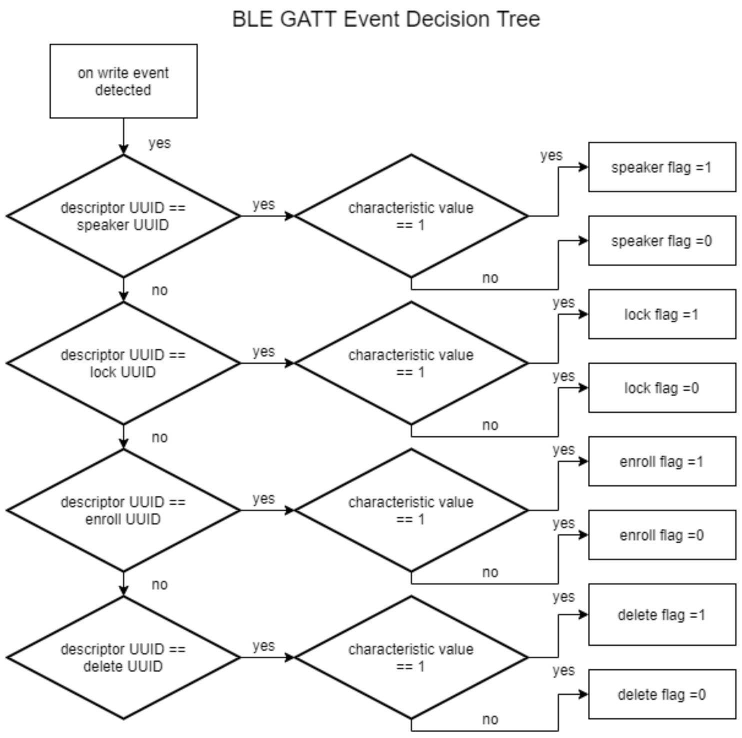
Subsystem Owner: Joseph Wang

Sub-system diagrams:



Subsystem Interaction:

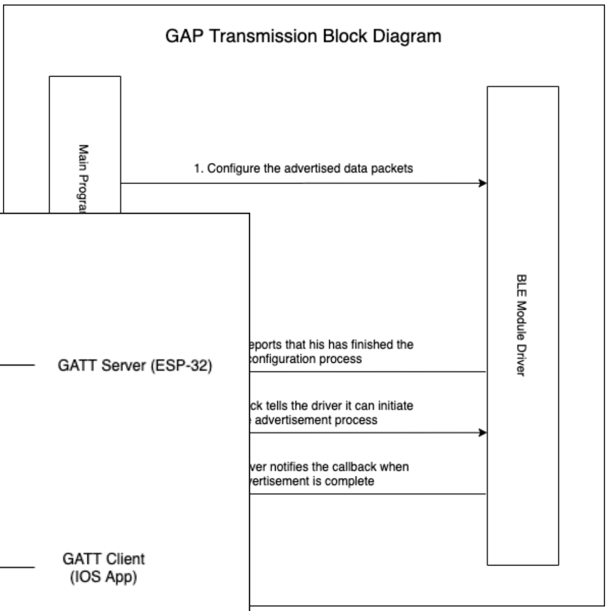
The BLE module will transmit real time information regarding the state of the lock to all peripherals it is connected to. The IOS app is subscribed to the GATT characteristic that stores the state of the lock meaning any changes to that state will be immediately known by the app. The speaker output is driven by the IOS app which leverages the BLE module to communicate with it. This means that when the IOS app is connected to the device, the user has the capability to interact with the speaker subsystem. The solenoid subsystem will also be driven by the BLE module as any changes in the state stored in the GATT characteristic will be reflected by the solenoids. (i.e. whether the solenoids should be opened or closed). The fingerprint scanner will simply provide a signal to

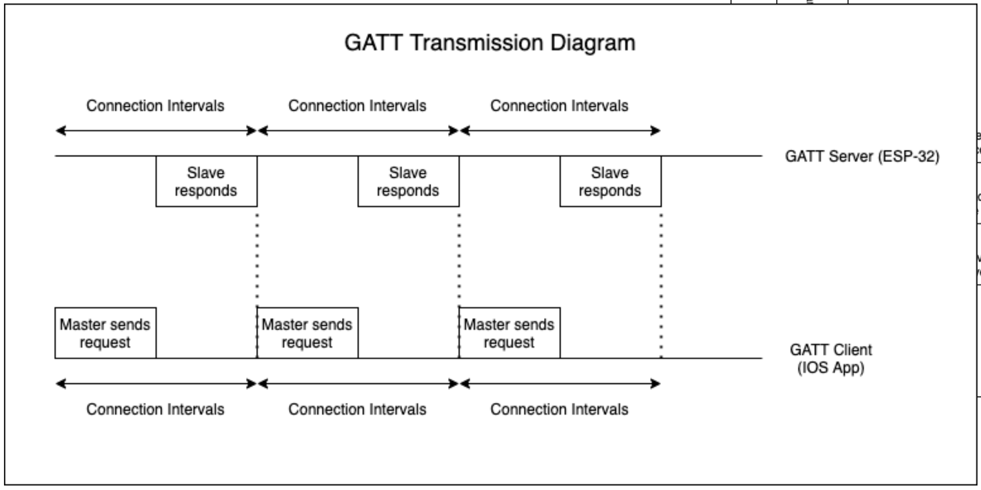

BLE GATT Event Decision Tree
on write event detected
yes
descriptor UUID == speaker UUID
yes
characteristic value == 1
yes
speaker flag =1
no
speaker flag =0
no
descriptor UUID == lock UUID
yes
characteristic value == 1
yes
lock flag =1
yes
no
lock flag =0
no
descriptor UUID == enroll UUID
yes
characteristic value == 1
yes
enroll flag =1
yes
no
enroll flag =0
no
descriptor UUID == delete UUID
yes
characteristic value == 1
yes
delete flag =1
yes
no
delete flag =0

the module as to whether or not a valid fingerprint was authenticated.

Theory of Operation:

When the user first downloads the app and opens up the lock, they will be forced to connect their Iphone to the lock via the onboard bluetooth module. The figure titled 'GAP Transmission' explains the fundamentals behind data advertisement of the BLE module. Upon the initial connection, the user will need to enter in a specific pin to confirm that it is pairing to the correct device (This pin is used as the encryption key for the AES-256 cryptographic algorithm that ensures the packet data transmitted is secure). Once the pairing and authentication has completed, the user's phone mac address is added to the server whitelist. On top of that, the user now has full access to the server's unlock_lock service, which allows them to unlock and lock the phone (locking and unlocking the phone is done by writing either a '1' or a '0'. Furthermore, they also have access to the onboard speaker if they wish to locate their bike through sound. The interaction between the BLE and the IOS Application is depicted in the figure titled 'Gatt Transmission Diagram'


GAP Transmission Block Diagram
Main Progra
1. Configure the advertised data packets
BLE Module Driver
eports that his has finished the
configuration process
ck tells the driver it can initiate
advertisement process
ver notifies the callback when
ertisement is complete



On the backend side, where the user has no access, the fingerprint scanner and the motor controller will first need to be processed through the onboard GPIO/Serial Pins before they can access the information stored on the bluetooth. Once the information has been processed, inputs are sent to the appropriate subsystem allowing the intended feature to be utilized.

Requirement	Specification	Justification
The GPS system of the user's phone must be able to connect to the bike lock in a bluetooth range of 15 feet	The BLE server on the ESP32 chip was programmed to have a transmission power level of 9 dB. According to the ESP 32 official documentation, setting a transmission power of 9 dB creates a bubble with a radius of around 5 m (16 ft) where connectivity fails.	The justification behind this design is to give the user an adequate amount of space to connect to the on board BLE server. It would be inconvenient to the user to have to be extremely close in order to connect to the bluetooth. service. This allows the user the capability to connect to their lock without being in visual sight in the case they wish to ping the onboard speaker.
Must reject non-user phones using our app	The BLE server is set up so that only one device can connect to it at a time. Furthermore, once a device has successfully cached the bonded connection requirements, that device becomes the only peripheral that is allowed past the BLE server's whitelist.	This requirement is justified by the fact that the user wouldn't want another user to be able to connect to their lock with their app and unlock their bike.

Core ECE design tasks

ECE362 - Microprocessor systems and interfacing

1. Leverage a microcontroller's GPIO, SIO, and Serial Protocols to interface with other subsystems.
    a. i.e. interfacing the ESP32 with the fingerprint scanner via UART, the speaker system with the DAC, and the motor controller via GPIO
2. Look through documentation and online resources for guidance to implement a BLE server on the ESP32

Schematics / parts / algorithm

The main part we will need for this subsystem is the BLE module built into the ESP32-Wroom-32D microprocessor. This subsystem requires nothing to be wired up, and is used simply by flashing source code to the processor that the chip sits on.

Standards

1. The BLE module will make use of the Bluetooth low energy protocol that is separate from normal Bluetooth 5. The standards for BLE are set by Bluetooth Special Interest Group.

Integration Plan:

Our integration plan's first step began when Ty and Joseph worked together on developing the IOS Application and BLE module. This design phase was implemented and tested modularly, meaning that any new feature changes would subsequently result in testing. The testing that was conducted consisted of being able to connect to the BLE module, being able to read and write data from specific characteristics, and being able to subscribe to services which allowed the IOS application to be notified of state changes on the BLE module.

Once the integration between the Iphone app and the BLE module was complete, the next step was to integrate all of the other subsystems together. The final design of the BLE module made integration relatively easy. This is due to the call back function called onWrite() provided by the NimBLE framework we used, which processed all the events regarding writing a value to any of the BLE's service characteristics. All hardware subsystems were designated a service in the BLE's GATT profile. Whenever a write event was detected, the onWrite() callback method was executed to handle this event by looking for the service it was trying to write to and the value it wished to write. The callback method would then compare that value with static values that indicated a certain process to be executed and set the appropriate flags that triggered the desired feature to activate. In order for those features to activate as close to real time as possible, a separate threaded process was implemented to constantly check those flag values for changes. If the threaded process detected a change, it would handle those changes by sending the appropriate commands to the appropriate subsystem. This method of using a flag system with a multi threaded process allowed us to implement complete system integration. One important aspect to note is that, aside from wiringthe subsystems to the microcontroller, most of the integration processes were done in software.

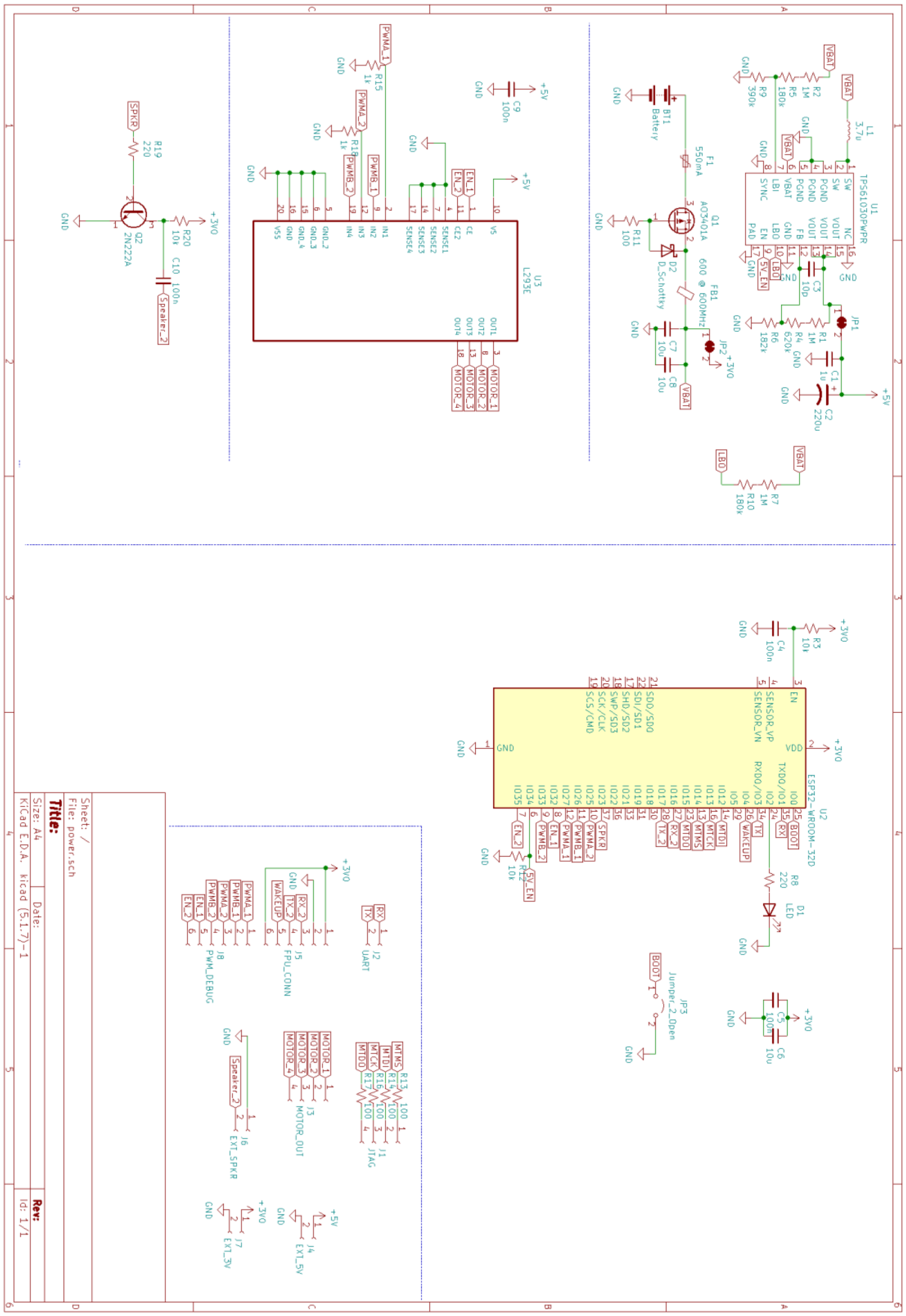

U1 TPS61030PWPR
L1 3.7u
VBAT
R2 1M
R5 180k
R9 390k
C1 1u
C2 220u
C3 10p
R1 1M
R4 620k
R6 182k
JP1
+5V
LBO
5V_EN
R7 1M
R10 180k
BT1 Battery
F1 550mA
Q1 AO3401A
R11 100
D2 D_Schottky
FB1 600 @ 600MHz
C7 10u
C8 10u
JP2
+3V0
U3 L293E
C9 100n
PWMA_1
PWMA_2
PWMB_1
PWMB_2
R15 1k
R18 1k
EN_1
EN_2
MOTOR_1
MOTOR_2
MOTOR_3
MOTOR_4
SPKR
R19 220
Q2 2N2222A
R20 10k
C10 100n
Speaker_2
U2 ESP32-WROOM-32D
R3 10k
C4 100n
R8 220
D1 LED
BOOT
RX
TX
WAKEUP
MTDI
MTCK
MTMS
MTDO
RX_2
TX_2
R12 10k
C5 100n
C6 10u
JP3 Jumper_2_Open
J1 JTAG
R13 100
R14 100
R16 100
R17 100
J2 UART
J3 MOTOR_OUT
J4 EXT_5V
J5 FPU_CONN
J6 EXT_SPKR
J7 EXT_3V
J8 PWM_DEBUG
GND
Sheet: /
File: power.sch
Title:
Size: A4
Date:
KiCad E.D.A. kicad (5.1.7)-1
Rev:
Id: 1/1

Figure: PCB Schematic

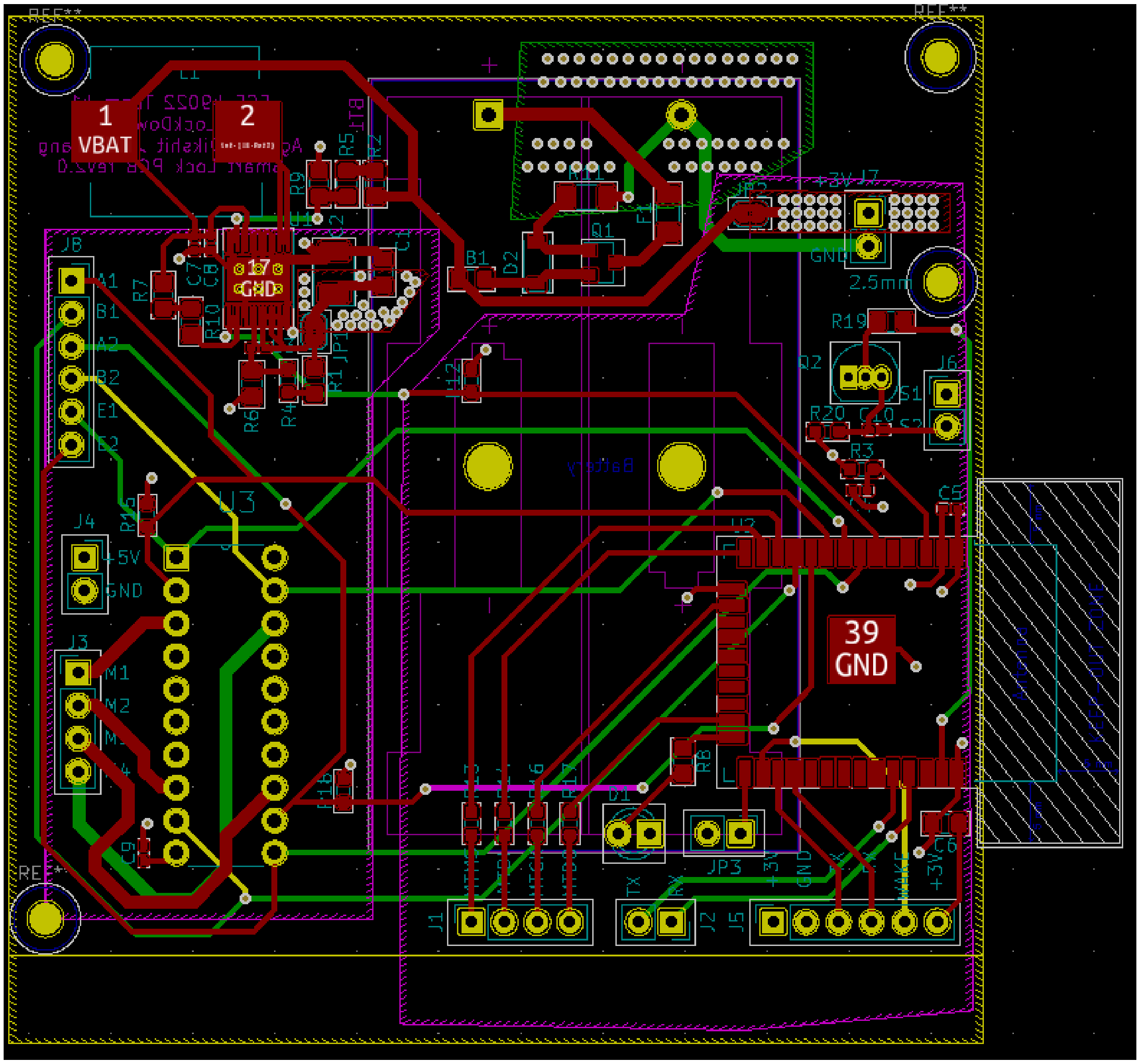


Figure: PCB Layout

Final Status of Requirements

1. **Must lock/unlock from user's fingerprint**

   <u>Met</u> - push and pull solenoids are used for the locking mechanism. Whenever an enrolled fingerprint is used, the solenoids circuit performs the locking mechanism.

2. **Must unlock from user's phone using our app if the fingerprint scanner fails**

Met - Pressing the lock and unlock button on the iOS app powers the solenoid circuit that locks/unlocks.

3. **Must reject non-user phones using our app**
   Met - The BLE server is set up so that only one device is able to pair with it at a time. Furthermore, we have set it up to show that once the device has bonded with it, it will be the only device that will be able to pair with the bluetooth module.

4. **Must reject non-user fingerprint**
   Met - The lock rejects fingerprints that are not enrolled using the iOS app.

5. **Must use the phone's GPS system in case the user forgets where the bike is located**
   Met - application's map drop a pin at the user's current location after pressing the "lock" button. The application is then closed and reopened followed by a press of the "Find My LockDown" button. This leads to the last location stored within the iPhone's persistent storage to be loaded and added to the map as a pin.

6. **The app must be able to store lock's GPS address.**
   Met - Every Time the lock gets locked, whether this be from the app or from the user manually locking the lock, the phone app will cache the GPS location. The phone app is able to know the current state of the lock because it is subscribed to receive a notification upon a change from unlocked to locked state from the ESP-32 BLE module

7. **The GPS system of the user's phone must be able to connect to the bike lock in a bluetooth range of 15 feet**
   Met - The BLE server was programmed to have a transmission energy level of 9 db. According to the documentation for the ESP 32 chip, a transmission power of 9 db sets the radius where connectivity is possible to around 5 meters (~16 feet). Therefore, the range is slightly larger than previously stated.

8. **Must lock from only the user's fingerprint.**
   Met - the lock only works when a fingerprint is enrolled in the sensor

9. **Must lock from user's phone using our app if the fingerprint scanner fails**
   Met - The purpose of the IOS application is to act as a fail safe for when the fingerprint scanner fails. Our video clearly demonstrates that the phone has an unlock and lock button that toggles depending on the current state of our lock. This button is successfully able to change the value stored on the ESP 32 module which in turn will be able to lock the user's phone.

10. **The app must be able to connect to speaker using bluetooth**
    Met - The main purpose of the speaker is to connect to the app via bluetooth. Whenever the find my bike button on the app is pressed, and the esp32 and app are connected through bluetooth, the speaker goes off to let the user know where the bike is. To demonstrate that I made the speaker available to connect to a mobile phone using the bluetooth and played an audio file.

Work Cited

[1] “LINKA Smart Bike Lock - Lock Smarter, Not Harder,” LINKA Smart Locks. [Online]. Available: https://www.linkalock.com/. [Accessed: 01-Sep-2020].

[2] Texas Instruments, “L293x Quadruple Half-H Drivers,” L293D datasheet, Sept. 1996 [Revised Jan. 2016].

[3] Adafruit, “Adafruit-Fingerprint-Sensor-Library”. [Online]. Available: https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library. [Accessed: 10-Oct-2020].

[4] Veeravalli, Anand. “Introduction to Low Dropout (LDO) Linear Voltage Regulators.” Design And Reuse, www.design-reuse.com/articles/42191/low-dropout-ldo-linear-voltage-regulators.html. Accessed 2 Dec. 2020.