ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Self-Charging Automatic Bike Lock

Team #18

PAUL JEONG (paulj3@illinois.edu)
JAKE LI (jli301@illinois.edu
RITHIK MORUSUPALLI
(rmorus2@illinois.edu)

TA: Jason Zhang

December 06, 2023

Abstract

The purpose of this document is to examine the creation of the Self-Charging Automatic Bike Lock. This device consists of a bike lock with Bluetooth functionality, an alarm, and a self-charging system. Bluetooth is a second-step verification and convenient feature, unlike other bike locks. The alarm is an extra measure of security to alert bystanders and ward off thieves, much like car alarms. The device charges the battery by the bicycle's movement and is another convenient and cost-effective feature.

Contents

1 Introduction

1.1 Purpose

According to bikelab.shop, it is estimated that 856 to 1070 bikes are stolen per year in the Champaign-Urbana area [1]. This underscores the importance of implementing security measures, such as bike locks and alarms. Nevertheless, traditional bike locks are susceptible to mechanical tampering, making them vulnerable to theft. Additionally, a significant portion of bike alarms necessitates the use of a remote for activation and deactivation.

Our solution is an automatic bike lock with an alarm system. It has both a physical locking mechanism and an alarm component. The physical lock consists of a linear actuator that inserts itself through the wheel spokes thus preventing motion, a keypad whose input changes the linear actuator's state, and a small speaker in conjunction with an accelerometer as the alarm component.

The device also contains a self-charging subsystem. This is achieved by attaching a friction generator to the back wheel of the bike and connecting it to the device's battery. To satisfy the requirements of our project, we designed a PCB using a microcontroller. We chose a microcontroller with Bluetooth functionality to implement our locking mechanism. To lock and unlock the device, it has to be paired to the user's phone with Bluetooth. The user can then press the keypad to lock or unlock the device. If the Bluetooth device is inaccessible, the user must input the correct sequence into the keypad.

1.2 High Level Requirements & Functionality

There are four high-level requirements for our project:

- The Bluetooth range must be limited to about a 1 meter radius from the device, resulting in the ability to change the state of the device by touching any one of the keys on the keypad.
- The self-charging system must be able to generate 6 watts and be able to power the device through multiple trips. The largest power consumption is from the linear actuator, and it must be met by the supply from the generator. This is to ensure the device is fully self-charging and not depleting.
- If there is not an established Bluetooth connection to the device, the device's state can be changed by inputting the correct pattern into the keypad. There must be an alternative option to changing the locked and unlocked state in the case when the user's Bluetooth device is inaccessible.
- When in the locked state, the linear actuator should extend to its full length of 4 inches and prevent the front wheel from spinning, thus preventing someone from stealing the bike.

1.3 Subsystem Overview

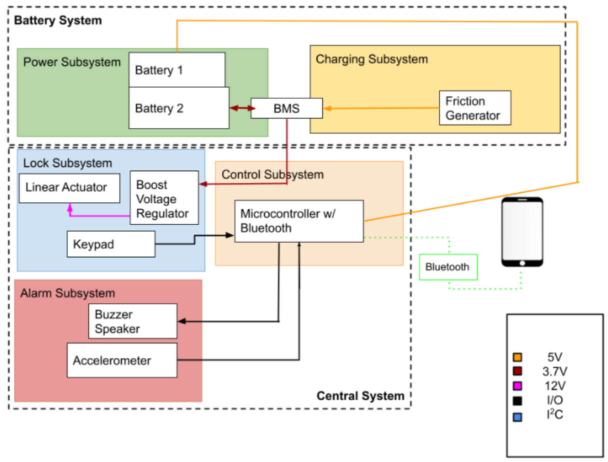


Figure 1: Final Block Diagram

1.3.1 Control Subsystem

The control subsystem controls the overall interactions among all subsystems. The control subsystem contains the Bluetooth-capable STM32WB55RG [2] microcontroller. If a Bluetooth connection from the user's phone to the device is established, a tap on the pound key of the keypad is required to change the state of the lock and disable the alarm subsystem. If there is no Bluetooth connection, then the state of the lock can still be changed with the input of the correct password into the keypad, which we set to "123". This logic is also handled in the locking subsystem.

1.3.2 Charging Subsystem

Our charging subsystem features the G.I. Bicycle Steel Dynamo [3] friction generator, which is attached to the side of the back wheel, as well as the Adafruit USB LiIon/LiPoly Battery Management System (BMS) [4]. As the back wheel spins, the friction generator will spin as well. The generated power will be sent into our BMS. This is done to regulate the voltage before charging the Adafruit Lithium Ion Battery (3.7V 6600mAh) [5] directly.

1.3.3 Power Subsystem

The power subsystem features two batteries, the Adafruit battery, and the Anker 321 Power Bank [6]. The Adafruit Battery is routed through the Taidacent 3.7 V to 12 V Boost Converter [7] to power the DC HOUSE Mini Electric Linear Actuator [8]. Since the battery supplies 3.7 V and the linear actuator requires 12 V, we use the step-up converter to boost our voltage to an acceptable range. Thus, this boosted voltage acts as the power supply for the physical locking mechanism. The Anker power bank supplies power to our NUCLEO-68 Development Board, which in turn supplies signals and power to our alarm subsystem's components.

1.3.4 Locking Subsystem

The locking subsystem consists of the Adafruit Membrane 3x4 Matrix Keypad [9] and the previously mentioned boost voltage converter and linear actuator. The subsystem's function has two states, locked (in which the linear actuator is extended) and unlocked (in which the linear actuator is retracted). The locking subsystem's state is determined by the input data from the control subsystem as follows:

- If a Bluetooth connection is detected on the board, a tap on the pound key of the keypad is required to extend or retract the linear actuator.
- If a Bluetooth connection is not detected on the board, the linear actuator can still change states through the input of the correct password on the keypad, which we set to "123".

We differentiate based on Bluetooth connection status to account for ease of use and edge case fulfillment. We allow for convenient unlocking if the Bluetooth connection is detected to allow the user to change the state of the locking subsystem in a more timely manner. We also allow users to change the state of the locking subsystem without a Bluetooth connection to address cases in which the user does not have access to their Bluetooth device. We also set limitations in the software so the linear actuator won't change states when the bike is in motion.

1.3.5 Alarm Subsystem

The alarm subsystem consists of the Active Piezo Electric Buzzer Alarm [10] and the Adafruit ST LSM6DSOX Accelerometer [11]. This subsystem acts to draw attention to the fact that someone is attempting to steal or tamper with the bicycle. The alarm subsystem is fully dependent on the control subsystem for both power delivery as well as activation instructions. It also provides the control subsystem with acceleration readings. The logic behind the alarm subsystem is as follows:

- If no Bluetooth connection is recognized on the board and the linear actuator is in the locked state, the accelerometer will constantly poll for the vibration readings and send them to the microcontroller. If the polled data shows acceleration greater than our set acceleration threshold or for some time greater than our time threshold, the alarm subsystem will receive the command to activate the buzzer speaker from the control subsystem.
- We set thresholds on acceleration and time to differentiate between the general nudges and bumps a parked bicycle will inevitably receive versus attempts to steal the bicycle.

The buzzer speaker will then continuously sound for a set amount of time until its deactivation conditions are met.

2 Design

The physical locking mechanism consists of a linear actuator, which can extend to 4 inches and is located on the front wheel of the bike, and a keypad, which is located in the center of the bike. The accelerometer, which is used as part of the alarm subsystem and connected to the control subsystem, is located within the PCB enclosure. The friction generator, which is part of the charging subsystem and connected to the BMS, is located on the back wheel of the bike.

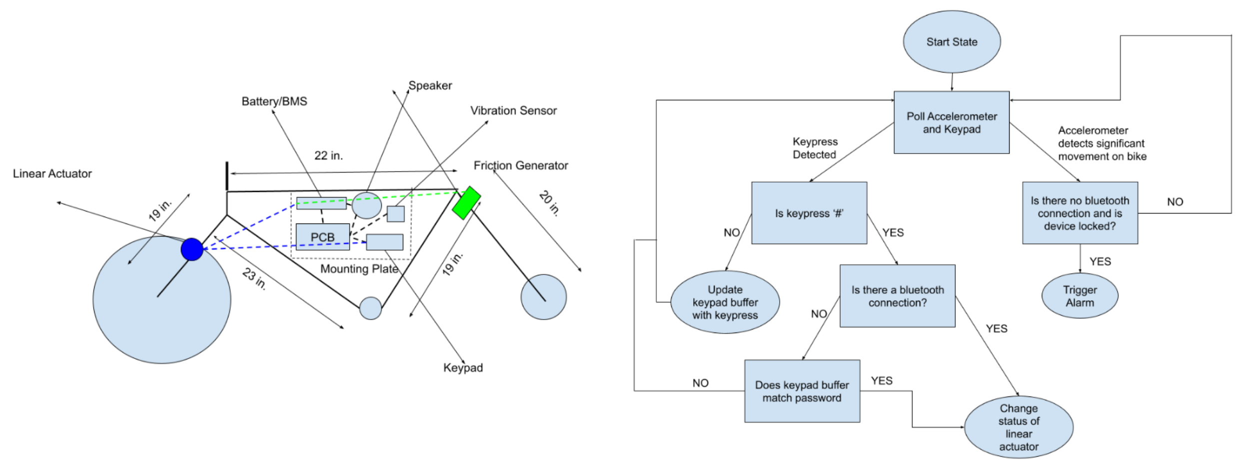


Figure 2: Physical Diagram and Software State Diagram

2.1 Control Subsystem Design

2.1.1 PCB with STM32WB55RG

Our initial PCB design was based on the NUCLEO-68 Development Board, we attempted to do a one-to-one recreation of the board as we thought that it would allow us full utilization of the board's features should we need it. We found routing and footprint/component size to be extremely difficult because even after removing the on-board ST-LINK/V2-1 debugger/programmer and converting to 4-layers. The 4-layers were also necessary for isolating Bluetooth/RF signals.

The next design was built on the STM32 Lora Router example and from there integrated the SMA Connector, accelerometer, and battery management system module. However, since the STM32WB55 pads have such little clearance, attempts to reflow and hand soldering caused many issues.

2.1.2 NUCLEO-68 Development Board

The STM32WB55RG [2] and Nucleo-68 Development Board were chosen because of their integrated Bluetooth capabilities. They also offer multiple external power input options and power mode options, which we would take advantage of considering the need for a long battery life.

2.2 Power Subsystem Design

2.2.1 Adafruit Battery Pack

We initially planned to use the 3.7V 6600mAh Adafruit Lithium Ion Battery [5] to power our PCB which would power all modules within our system. However, once we recognized that our PCB did not function properly before our final deadline we decided to use the Nucleo-68 Development Board as the primary component of the control subsystem. To power the development board with an external battery such as our battery pack, we had to make changes to the Nucleo-68 Development board. This consisted of changing some of the solder jumpers on the board (specifically closing Solder Jumper 26 and opening Solder Jumper 27). This in theory should have allowed us to fully power our Nucleo-68 board through the battery since it is set to accept 1.8V-3.3V.

However, once we implemented the soldering changes to the development board we noticed that while we were successfully powering the microcontroller on the board, since we were able to make a Bluetooth connection, we did not receive any power from the GPIO pins on the Nucleo-68 Development Board. After multiple attempts to rectify this issue, including re-soldering the specified jumpers, we decided to utilize the battery pack only for our linear actuator with the help of a boost converter. We then chose to add a new battery to our system, the Anker 321 Power Bank, to power the development board. Since the power bank has a USB, it can be directly connected to the development board, which successfully provides power.

Our current battery life for the Adafruit Battery Pack in its current power configuration with the Linear Actuator (constantly powering it) can be calculated as:

$$Battery\ Life = \frac{Battery\ Capacity(maH)}{Load\ Current(mA)} = \frac{6600mAh}{500mA} = 13.2\ hours \tag{1}$$

2.2.2 Anker 321 Power Bank

The Anker 321 Power Bank [6] is used to provide power to our Nucleo-68 Development Board. Initially not part of our overall system, the Anker power bank is an alternative to our initial power delivery method. Due to the nonfunctional PCB, we chose to use the NUCLEO-68 to successfully meet the high-level requirements listed. Since testing showed that it was not possible to power the GPIO pins of the development board using the Adafruit battery pack, we pivoted to power the Nucleo-68 Development Board using the Anker power bank. Delivering 5 V, 5200 mAh to the board, the power bank can fully

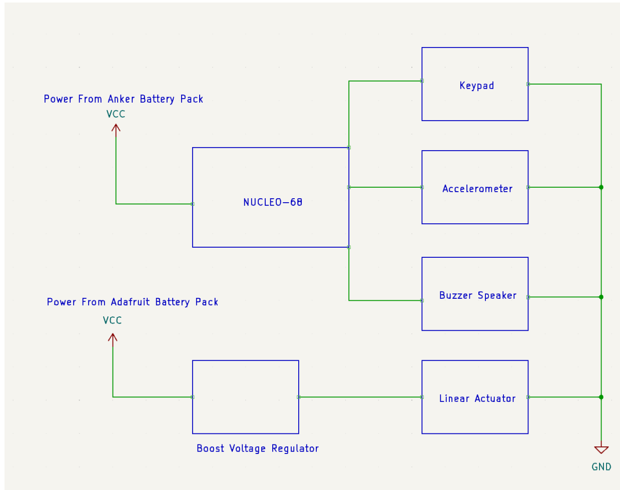


Figure 3: Schematic Showing Overall Power Delivery

power both the board and all of the GPIO outputs so full completion of the high-level requirements are met.

2.3 Charging Subsystem Design

2.3.1 G.I. Dynamo Friction Generator

The G.I. Dynamo Friction Generator [3] is the primary component of the charging subsystem that charges the Adafruit battery pack without the use of external power sources. As stated in the charging subsystem overview, the friction generator functions by coming into contact with the spinning back wheel. Inside the dynamo, there are coils of wire and a magnet. As the dynamo rotates, the magnetic field around the magnet changes relative to the coils of wire. The changing magnetic field induces an electromotive force (EMF), and then the induced EMF creates an electrical current in the wire coils. This electrical current is then delivered to our battery management system.

An alternative method of recharging the battery pack was the use of solar cells. However, due to solar cells' dependencies on current weather as well as placement relative to sunshine, we chose to not go with this option. In these cases, the cell would not efficiently recharge our battery pack.

2.3.2 Battery Management System

The Adafruit USB LiIon/LiPoly Battery Management System [4] intakes linear or unsteady power delivery and ensures linear charging to the battery. Since the friction generator is dependent on the cycling motion, its power output varies, which is why the BMS is necessary. It also mediates discharging from the battery to the load, which the linear actuator and NUCLEO-68 Development Board require. Also, the BMS has a USB-Micro port as an alternative charging option for when the bike is locked and dead.

Initially, the BMS was intended to be integrated within the PCB. Due to our PCB malfunctioning, we utilized the BMS module externally. With the proper configurations set for the NUCLEO-68 Development Board to accept 1.8 V-3.3 V external power input, this should have functioned, however, only the Bluetooth was operable and no GPIO functionality was. This led to a change in design for the BMS to only power the linear actuator while the NUCLEO-68 Development Board would be powered by another battery pack separate from the BMS.

The math behind this is as follows:

$$P_{friction\ generator} = V_{friction\ generator} * I_{friction\ generator} = 12V * 0.5A = 6\ W \tag{2}$$

$$P_{BMS} = V_{BMS_{Max}} * I_{BMS_{Max}} = 5V * 0.5A = 2.5W \tag{3}$$

This should have meant that friction generator would provide too much voltage to our BMS, however as we further explain in our Design Verification section, this ended up working quite well for us.

2.4 Locking Subsystem Design

2.4.1 Linear Actuator

The DC HOUSE Mini Electric Linear Actuator [8] was used as our physical locking mechanism, as it would prevent the bike from rolling forward if the linear actuator was extended. It also would not retract if someone happened to cut the wires that connect the linear actuator to the rest of the device, since linear actuators need to be supplied some amount of voltage and current to extend or retract. With this particular linear actuator, it needed +12 V to extend and -12 V to retract, necessitating the need for a relay (discussed in 2.4.4). Since the Adafruit Battery Pack supplies 3.7 V, a boost voltage regulator is needed to boost it to 12 V for the linear actuator (discussed in 2.4.2).

2.4.2 Boost Voltage Regulator

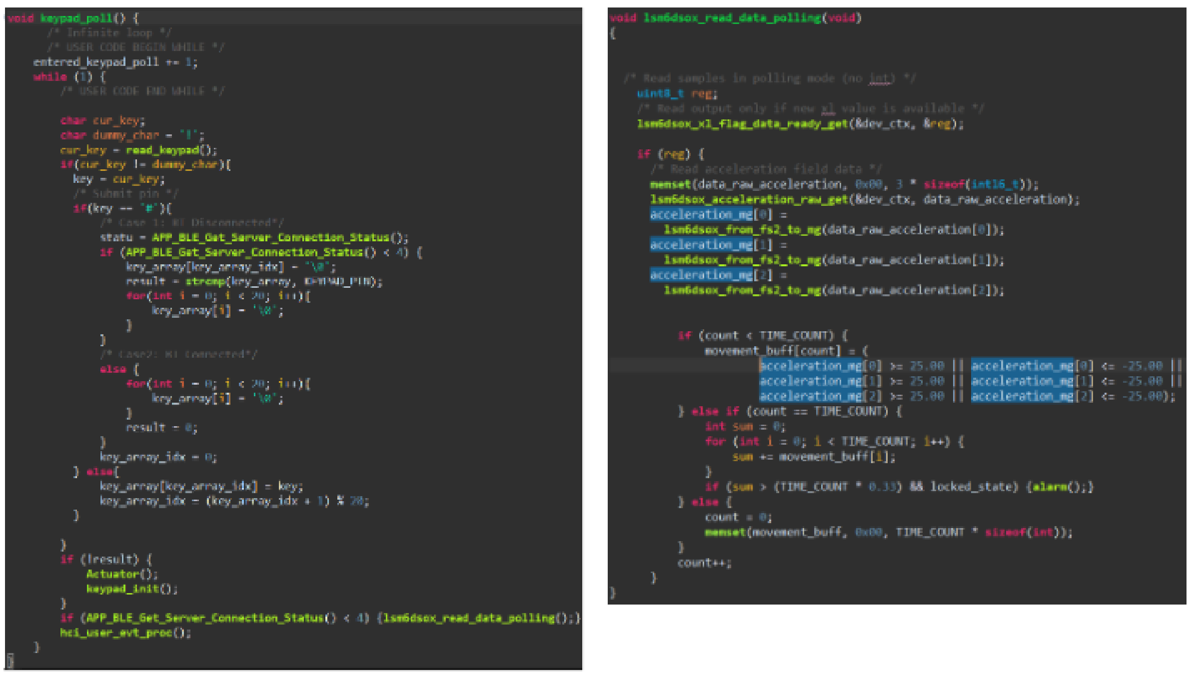
The Taidacent 3.7V to 12V DC-DC Boost Step Up Converter [7] would boost the voltage from the battery to the level needed to extend or retract the linear actuator. We considered using a 12 V battery and directly powering the linear actuator using it, but we were unable to find a suitable battery charger for a 12 V battery that would fit within our design.

2.4.3 Keypad

The Adafruit Membrane 3x4 Matrix Keypad [9] was chosen for two reasons. One, it would not require any extra mechanical work from the Machine Shop. Two, there was extensive documentation on how to interface this specific keypad with the STM32 Microcontroller.

Initially, we were planning on using an array of touch sensors as user input. However, we learned from the Machine Shop that attaching an array of touch sensors to the bike would require time-consuming assembly on their part. As a result, we pivoted to our alternative, that being the keypad. The keypad also has an adhesive back, which would allow easy installation.

We initially planned on interfacing the keypad with our microcontroller using an interrupt-based approach that would use the built-in EXTI interrupts within the STM32. This approach ended up being very buggy, as it would sometimes register a single keypress multiple times or register a single keypress as a random sequence of keypresses. We switched to a polling-based approach, in which the keypad would be repeatedly polled for key presses. This approach ended up being reliable and we were able to integrate it with the Bluetooth and the accelerometer.



```
void keypad_poll() {
    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
  entered_keypad_poll += 1;
  while (1) {
      /* USER CODE END WHILE */

      char cur_key;
      char dummy_char = '!';
      cur_key = read_keypad();
      if(cur_key != dummy_char){
        key = cur_key;
        /* Submit pin */
        if(key == '#'){
          /* Case 1: BT Disconnected*/
          statu = APP_BLE_Get_Server_Connection_Status();
          if (APP_BLE_Get_Server_Connection_Status() < 4) {
            key_array[key_array_idx] = '\0';
            result = strcmp(key_array, KEYPAD_PIN);
            for(int i = 0; i < 20; i++){
              key_array[i] = '\0';
            }
          }
          /* Case2: BT Connected*/
          else {
            for(int i = 0; i < 20; i++){
              key_array[i] = '\0';
            }
            result = 0;
          }
          key_array_idx = 0;
        } else{
          key_array[key_array_idx] = key;
          key_array_idx = (key_array_idx + 1) % 20;
        }

      }
      if (!result) {
        Actuator();
        keypad_init();
      }
      if (APP_BLE_Get_Server_Connection_Status() < 4) {lsm6dsox_read_data_polling();}
      hci_user_evt_proc();
  }
}
```

```
void lsm6dsox_read_data_polling(void)
{


  /* Read samples in polling mode (no int) */
    uint8_t reg;
    /* Read output only if new xl value is available */
    lsm6dsox_xl_flag_data_ready_get(&dev_ctx, &reg);

    if (reg) {
      /* Read acceleration field data */
      memset(data_raw_acceleration, 0x00, 3 * sizeof(int16_t));
      lsm6dsox_acceleration_raw_get(&dev_ctx, data_raw_acceleration);
      acceleration_mg[0] =
        lsm6dsox_from_fs2_to_mg(data_raw_acceleration[0]);
      acceleration_mg[1] =
        lsm6dsox_from_fs2_to_mg(data_raw_acceleration[1]);
      acceleration_mg[2] =
        lsm6dsox_from_fs2_to_mg(data_raw_acceleration[2]);


        if (count < TIME_COUNT) {
            movement_buff[count] = (
                    acceleration_mg[0] >= 25.00 || acceleration_mg[0] <= -25.00 ||
                    acceleration_mg[1] >= 25.00 || acceleration_mg[1] <= -25.00 ||
                    acceleration_mg[2] >= 25.00 || acceleration_mg[2] <= -25.00);
        } else if (count == TIME_COUNT) {
            int sum = 0;
            for (int i = 0; i < TIME_COUNT; i++) {
                sum += movement_buff[i];
            }
            if (sum > (TIME_COUNT * 0.33) && locked_state) {alarm();}
        } else {
            count = 0;
            memset(movement_buff, 0x00, TIME_COUNT * sizeof(int));
        }
        count++;
    }
}
```

Figure 4: Images of polling code

2.4.4 Locking Circuit

The locking circuit consists of the boost voltage regulator, an AGQ26024 12 V DPDT relay [12], and an IRF510 Power MOSFET [13]. The boost voltage regulator (discussed in 2.4.2) is connected to our Adafruit battery, with the output connected to the Power MOSFET. The gate input for the MOSFET was connected to the microcontroller, signifying when the device should change its state (locked-¿unlocked or unlocked-¿locked). The source pin of the MOSFET was connected to the polarity pins of the DPDT relay, with the output of the relay connected to our linear actuator. The DPDT relay would alternate between outputting a positive 12 V and a negative 12 V, with its output being connected to the linear actuator.

2.5 Alarm Subsystem Design

2.5.1 Accelerometer

The accelerometer we used was the Adafruit LSM6DSOX [11]. This module has acceleration reading capabilities of up to 16 g. For vibration detection, 2 g is enough. Initially, we planned to use a vibration sensor, to implement theft detection. A vibration sensor reports the frequency of vibration signals. However, an accelerometer offers tangible values that can be programmed to be more of a dynamic response and sensitive to motion patterns.

A physical consideration was that the accelerometer must be free-hanging within the enclosure for precise readings. If secured to the enclosure, then the accelerometer will deliver minimal data. A higher range of data values is preferred because it is easier to distinguish movement and create thresholds. The accelerometer was connected by providing 3.3 V and connecting I2C SCL (serial clock) and I2C SDA (serial data) to the STM32.

Utilizing the accelerometer's drivers and read functions, we conducted polling to continuously read the acceleration data. However, due to the keypad also polling without a multi-tasking implementation, we decided to incorporate the accelerometer's polling within the keypad's polling. After formatting the data for X, Y, and Z directions, we created thresholds based on intensity and duration. Due to our testing, we settled on 25 mg (milli-gravity) as our intensity lower limit and 33% of 128 ms as the duration lower limit condition. This provided the best results when testing scenarios for accidental bumping versus shaking and picking up the bike.

2.5.2 Buzzer Speaker

The buzzer speaker we used was the Piezo Electronic Buzzer [10]. It accepts 3-24 V and is rated at 12 V for 100 dB, which is more than enough for our needs. We supplied 3.3 V through a GPIO output and received 87 dB, which is around the same loudness as heavy traffic. Initially, we were not aware of buzzers and only looked at speakers, which require a signal waveform for sound production. The buzzer is much simpler and serves our purpose.

3 Design Verification

3.1 Control Subsystem Verification

3.1.1 Bluetooth Range

As stated in the Control Subsystems R&V Table, the Bluetooth connection must have a reasonable range and must ignore the user's Bluetooth device until the bicycle is within arm's reach. The Bluetooth range must be minimal to correctly disconnect and activate the lock and alarm subsystem, but also large enough for the case when the user has their phone in a backpack or wants to wheel their bicycle. The requirement for the Bluetooth range was set to 1 meter.

The test was conducted in a hallway with no physical obstructions. Starting within Bluetooth range, we slowly walked away from the device. This approach was done because when attempting to start out of range and walk into range while attempting to connect would have instances where it would momentarily connect and then immediately disconnect. The test concludes with an average of 2.32 meters as the Bluetooth range of the

Trial Number	**Bluetooth Range(Meters)**
1	2.5
2	2.8
3	2.5
4	2
5	1.8

Table 1: Bluetooth Test Results

device. Through software, we have given the lowest TX power level to Bluetooth, which gives us the shortest Bluetooth range by code. This reduced the rated 240 meters by Bluetooth 5.0 on the NUCLEO-68 Development Board to 2 meters. We could have added physical restrictions to the Bluetooth range to shorten the range further, however, we ran out of time. To note, although 2 meters does not exactly meet our full expectations, it is still a reasonable range for users and will not hinder the performance of our bike lock as stated in the Control Subsystems R&V Table. In this sense, the requirements were only partially met.

3.2 Power Subsystem Verification

3.2.1 Proper Power Delivery to the NUCLEO-68

Our initial requirement and verification with our power subsystem came from the proper power delivery from the battery pack to our PCB, which would then send power out of the PCB to our various components. As we approached the final demo, however, the only concern of power delivery in our power subsystem became the power from our Anker 321 Power Bank to our NUCLEO-68 Development Board. So our new requirement for this subsystem was the successful delivery of the power bank's 5 V to our NUCLEO-68 Development Board. There are multiple ways we can confirm the success of this requirement. We can both observe the states of the NUCLEO-68 Development Board's power LED indicators to confirm power delivered. We can also use the multimeter and manually check if 5V is being delivered to the board through its micro-USB port. We measured the voltage across the micro-USB pins multiple times and displayed them on the following plot:

Based on the data, we were able to successfully power our NUCLEO-68 Development Board with the necessary 5 V for proper operation. We could also confirm this power as the power indication LEDs were properly lit. Our original R&V table also stated that we must successfully provide 3.3-3.6 V to our microcontroller. However, as we transitioned to using the NUCLEO-68 Development Board, we did not have to do a verification check as the NUCLEO-68 Development Board has an inbuilt voltage step-down to ensure that this requirement is met without outside help.

3.3 Charging Subsystem Verification

3.3.1 Power From Friction Generator

Our friction generator is rated to deliver 6 W at a maximum of 12 V and 0.5 A at a time. We had a requirement to deliver this wattage to our BMS. The only way we could get testing data for the power generated from our friction generator would be to observe, using a multimeter, the voltage and current generated as the bicycle wheel spins. We did this by flipping the bike upside down and moving the pedals with our hands. We then connected the power and ground wires from the generator to a breadboard and then read the readouts from the board with a multimeter. Our measured data does have an initial ramp-up period in which the voltage and current readings are extremely low as we speed up the motion of the bike. Our data is as follows:

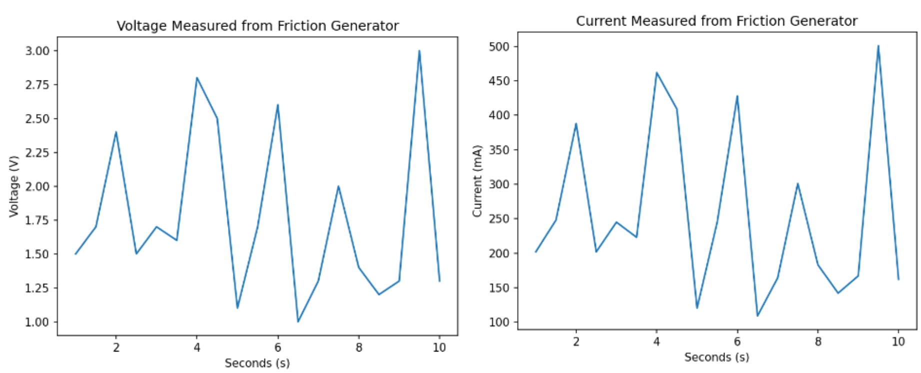


Figure 5: Voltage/Current Measured from Friction Generator

From our data, we see that the voltage and current readings from our friction generator are extremely variable as the voltage values jump between 2 to 3 V and the current values jump from 0.1 to 0.5 A. This means that we were never able to achieve the rated power delivery of 6 W as stated by our friction generators datasheet.

However, given that we used the Adafruit BMS module in our final design we also have to keep the power limitation it can intake in mind as well: our BMS module is rated to take a maximum of 2.5 W from the generator. Our data shows that we do not exceed approximately 0.5 A or 3 volts. As such, we do not exceed the BMS's capabilities and thus protect against burning out the BMS. Therefore, while we may have not met our originally stated verification, the power delivered from the friction generator does meet the power delivery needs we had in our final project assembly. We state that we partially passed our requirement and verification check for charging from the friction generator. As we do not meet our wattage goal, we do provide sufficient power to charge the battery at a safe level that will not burn out our BMS.

3.3.2 Battery Charging Module

The second entry in our R&V table regarding our charging subsystem concerns the functionality of our BMS. Given that we supply both the aforementioned inconsistent voltage

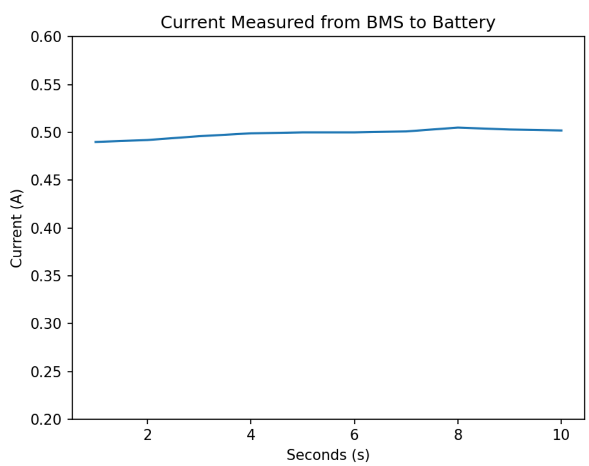


Figure 6: Current Measured from BMS to Battery

and current from our friction generator, our BMS has to be capable of taking in this inconsistent power and successfully supplying linear and consistent power to our Adafruit Battery Pack. To test this, we had to measure the output voltage and current from the out pins of our BMS module, to see if a steady current was within an acceptable range of 0.1 A to 0.5 A. Our data is shown in Figure 7. Through analysis of our data, we can see that the power delivered from the BMS to the battery is consistent (3.7 V, 0.50 A, 1.85 W). Thus we successfully provide linearized and consistent power to our battery and pass our battery charging module verification check.

3.4 Locking Subsystem Verification

3.4.1 Boost Voltage Regulation

The boost voltage regulator was expected to boost voltage from the battery from 3.7 V to 12 V, as stated in the first entry in the Locking Subsystem R&V Table. To test this, we measured the output voltage coming out of our boost voltage regulator for ten seconds. The resulting average voltage over the ten-second interval was 12.23 ± 0.2 V. This result satisfies the first entry in our R&V table for the Locking Subsystem.

3.4.2 Proper Keypad Functionality

The second entry in the Locking Subsystem R&V Table states that the keypad should function correctly whether or not there is a Bluetooth connection to the device. To verify this, we tested every single possible scenario for the keypad, checking both the contents of the keypad buffer and the state of the lock. The results from our verification matched

Scenario	**Results**
Any key besides '#' pressed	keypress added to keypad buffer, no change in state of lock
'#' key pressed and Bluetooth connection has been established	State of lock changed, keypad buffer reset
'#' key pressed, no Bluetooth connection, keypad buffer holds correct password of '123'	State of lock changed, keypad buffer reset
'#' key pressed, no Bluetooth connection, keypad buffer does not contain correct password	State of lock unchanged, keypad buffer reset

Table 2: Keypad Test Results

with the entry in the R&V table, as the "#" key would change the state of the lock if there was a Bluetooth connection, and the correct password was needed to change the state of the lock if there was no Bluetooth connection.

3.4.3 Physical Movement Testing

The linear actuator, when extended, should prevent the bike from rolling, as stated in the Locking Subsystem R&V Table. We tried rolling the bike both forward and backward with the linear actuator extended, and the bike was not able to roll.

3.5 Alarm Subsystem Verification

3.5.1 Proper Recognition of Negligible Movement

As stated in the Alarm Subsystem R&V Table, the device must be able to ignore slight movements. To simulate this scenario, a test was conducted by asking others to walk by and bump minimally into the bicycle. We used our best judgment to tweak the sensitivity of our alarm subsystem. With the final settings, the recognition of negligible contact was consistent and valid.

3.5.2 Proper Recognition of Substantial & Continuous Movement

The device must be able to recognize substantial and continuous movement. To test this scenario, we have simulated by asking others to shake and pick up the bike. We used our best judgment to tweak the sensitivity of our alarm subsystem.

With the final settings, the recognition of substantial and continuous movement was consistent and valid. During our test run, each time the alarm was triggered. With a success rate of 100%, the requirement of proper recognition of substantial and continuous movement was met.

4 Cost & Schedule

4.1 Project Costs

4.1.1 Parts Cost

The total cost of all parts for this project came out to $\$220.82$. Please refer to Appendix 7.1 to see the Breakdown of Costs.

4.1.2 Labor Cost

The average salary of a Computer Engineering graduate from the University of Illinois Urbana-Champaign is $109,176 per year as of 2021/2022 [14]. Assuming graduates are working eight hours per day, totaling 40 hours a week, for 52 weeks in a year, we calculate the total amount of hours worked in a year as 2080 hours. Dividing the annual salary by 2080 gives us our per-hour cost of $52.49. We find the number of weeks we worked on our project by subtracting the total amount of weeks in a semester (16) by the number of weeks it took for our project to be approved (3) and removing fall break. Our result is a total of 12 weeks. We can also assume that each individual will work a total of 12 hours per week on the 445 project. Having acquired all of this data, we can find our per-individual labor cost for the project:

$$LaborCost_{individual} == \frac{\$52.47}{1hour} * \frac{12hours}{Weeks_{number}} * 12Weeks_{number} = \$7555.68 \tag{4}$$

We then multiply our calculated per individual cost of $7555.68 by 3, and find the total labor cost of this project to be $22667.04 for our three-member team.

4.1.3 Total Cost

This brings our total cost for our project to $\$220.82 + \$22667.04 = \$22887.86$

4.2 Schedule

Week	Task	Team
Week of 9/18	Begin Design Document	Jake, Paul, and Rithik
Week of 9/25	Finalize Design Document	Jake, Paul, and Rithik
Week of 9/25	Begin initial PCB design in KiCad	Jake, Paul, and Rithik
Week of 10/02	Work on initial PCB design in KiCad	Jake, Paul, and Rithik
Week of 10/02	Attend Design Review	Jake, Paul, and Rithik
Week of 10/02	Search for bicycle to purchase	Rithik
Week of 10/09	Bicycle has been purchased and delivered to machine shop	Rithik
Week of 10/16	Second round PCBWay ordering met	Jake, Paul, and Rithik
Week of 10/23	Third round PCBWay ordering met	Jake, Paul, and Rithik
Week of 10/30	Begin ordering PCB Components and Breadboard the relay circuit	Jake and Paul
Week of 10/30	Breadboard the relay circuit	Jake and Paul
Week of 11/06	Continue circuitry and code	Jake and Paul
Week of 11/06	Finalize PCB component purchasing	Rithik and Jake

Week of 11/13	Finish breadboarding design and finalize code on development board	Jake and Paul
Week of 11/13	Begin soldering our final PCB	Rithik
Week of 11/20	Thanksgiving break	Jake, Paul, and Rithik
Week of 11/27	Finalize our design using the development board for the Final Demo	Jake, Paul, and Rithik
Week of 11/27	Final Demonstration	Jake, Paul, and Rithik
Week of 12/04	Prepare for and attend our Final Presentation and submit our final documents	Jake, Paul, and Rithik
Week of 12/04	Submit our final documents	Jake, Paul, and Rithik

5 Conclusion

5.1 Accomplishments

We partially fulfilled the high-level requirements regarding restricting Bluetooth range to ~2 meters, we also programmed the keypad to function independently of Bluetooth, incorporated a functional linear actuator as a physical lock, and integrated self-charging for the system. Ultimately, we achieved our goal by creating an automatic bike lock with self-charging capabilities and an integrated alarm system for enhanced security. The bike remained easy to ride with all components (with the locking mechanism, self-charging system, and alarm system) working properly.

5.2 Uncertainties

The main failure of our project was our inability to develop a working PCB design. Due to ordering issues and delays with board delivery, we were only able to begin board assembly less than a week before the final demo. After soldering the components onto the board, we discovered that we were unable to power or flash our microcontroller. We used the multimeter to check the pins of our micro-USB port from which the board should have been powered through, however, we were unable to detect the test voltage of 5 V on the pins instead measuring a negligible 0.001-0.002 V. We believe a more in-depth dive into the actual power traces of our board needs to be reviewed to find a potential issue.

We were able to limit the Bluetooth range from 240 meters to $\sim$2 meters by software, which is an acceptable range in terms of functionality. However, this does not meet our stated requirement of $\sim$1 meter.

We attempted to power the NUCLEO-68 Development Board with the 3.7 V battery pack, following the documentation given for the board on battery pack power. The NUCLEO-68 Development Board was able to be powered on, and we were able to connect to the NUCLEO-68 Development Board using Bluetooth. However, none of the GPIO pins registered any significant voltage. We believe that the issue may be related to the voltage of our battery pack, as the micro-USB port provides 5 V to the NUCLEO-68 Development Board. In retrospect, we could have used a 5 V battery instead of a 3.7 V battery, although we would likely have to make changes to our battery charging module.

5.3 Ethical Considerations

As members of Team 18, we pledge our unwavering commitment to upholding the highest standards of ethical and professional conduct throughout this project, guided by the principles outlined in section 7.8 of the IEEE Code of Ethics [15]. Our foremost priority is to prioritize the safety, well-being, and welfare of the public, particularly during the testing phases of our project. We are dedicated to ensuring transparency in all our testing activities, particularly those that may carry hypothetical risks to the public or the environment.

5.4 Future Works

To enhance the security and accessibility of the Self-charging Automatic Bike Lock, there are some ideas yet to be implemented. Convenience is something we would like to further address by creating a modular clamping design. As of now, our device is attached to the bicycle by a metal plate and mechanical attachments for the linear actuator and generator. We would like to do more research on clamping mechanisms that will allow for an easy installation, yet secure and irremovable system.

The alarm operation is another area where we would like to make improvements. As of now, when triggered, it will beep for about 10 seconds. To mimic car alarms, a requirement of a Bluetooth connection or correct PIN entry should be the only method of disabling the alarm.

6 References

References

[1] bikelab.shop, *Bike theft in champaign-urbana*, Available at https://bikelab.shop/blog/bike-theft-champaign-urbana (Accessed 9/11/23).

[2] ST, *Ultra-low-power dual core arm cortex-m4 mcu 64 mhz, cortex-m0+ 32 mhz with 256 kbytes of flash memory, bluetooth le 5.4, 802.15.4, zigbee, thread, usb, lcd, aes-256*, Avaliable at https://www.st.com/en/microcontrollers-microprocessors/stm32wb55rg.html (Accessed 9/14/23).

[3] Amazon, *G.i. bicycle steel dynamo 12 volts friction generator with fitting clamp for cycle enthusiasts and for diy projects*, Avaliable at https://www.amazon.com/GI-Friction-Generator-Enthusiasts-Projects/dp/B01CTBILWG/ref=sr_1_8?keywords=bicycle+generator&qid=1694734317&sr=8-8 (Accessed 9/14/23).

[4] adafruit, *Usb lilon/lipoly charger [v1.2]*, Avaliable at https://www.adafruit.com/product/259 (Accessed 10/20/23).

[5] adafruit, *Lithium ion battery - 3.7 v 6600mah*, Avaliable at https://www.adafruit.com/product/353 (Accessed 9/11/23).

[6] Amazon, *Anker power bank, 5,200 mah portable charger, compatible with iphone 15/15 plus/15 pro/15 pro max, iphone 14/13 series, samsung, google pixel, lg, and more (black)*, Avaliable at https://www.amazon.com/Anker-PowerCore-Ultra-Compact-Portable-Compatible/dp/B09NRG2YT3?th=1 (Accessed 12/6/23).

[7] Amazon, *Taidacent 10pcs 3.7v to 12v dc-dc boost step up converter boost converter module 3.7v lithium battery power booster 12v voltage regulator*, Avaliable at https://www.amazon.com/Taidacent-Converter-Lithium-Battery-Regulator/dp/B08J48TFVV (12/1/23).

[8] Amazon, *Dc house mini electric linear actuator stroke 4"–force 4.5 lbs–12v — high-speed 1.97"/sec–weight 0.25kg ideal for intelligent range hood, fan blades, cabinets, robotics, home automation*, Avaliable at https://www.amazon.com/Electric-High-Speed-sec-Weight-Intelligent-Automation/dp/B07ZJ46947/ref=cm_cr_arp_d_product_top?ie=UTF8&th=1 (Accessed 9/13/23).

[9] Amazon, *Adafruit membrane 3x4 matrix keypad + extras [ada419]*, Avaliable at https://www.amazon.com/Adafruit-Membrane-Matrix-Keypad-Extras/dp/B00NAY2XUS/ref=sr_1_4?keywords=keypad+arduino+3x4&qid=1701480110&sr=8-4 (12/1/23).

[10] Amazon, *6 pcs active piezo electric buzzer alarm, dc 3-24v low voltage buzzer beep tone speaker alarm with lead wire for classroom, game show, physics circuits intermittent sound*, Avaliable at https://www.amazon.com/Electric-Voltage-Classroom-Circuits-Intermittent/dp/B07SQ8GDPZ (Accessed 12/01/23).

[11] adafruit, *Adafruit lsm6dsox 6 dof accelerometer and gyroscope - stemma qt / qwiic*, Avaliable at https://www.adafruit.com/product/4438 (Accessed 12/01/23).

[12] digikey, *Agq20012*, Avaliable at https://www.digikey.com/en/products/detail/panasonic-electric-works/AGQ20012/510127 (Accessed 12/01/23).

[13] vishay, *Irf510*, Avaliable at https://www.vishay.com/docs/91015/irf510.pdf (Accessed 12/01/23).

[14] U. of Illinois Urbana-Champaign, *Salary averages*, Avaliable at https://ece.illinois.edu/admissions/why-ece/salary-averages (Accessed 9/14/23).

[15] ieee.org, *Ieee code of ethics*, Avaliable at https://www.ieee.org/about/corporate/governance/p7-8.html (Accessed 9/14/23).

7 Appendix

7.1 Parts Cost Table

Part Name	Part Description	Vendor	Quantity	Price per Unit($)	Total Price($)
P-NUCLEO-WB55	Dev Board	ST	2	43.09	100.45
USB LI-ION/LIPOLY	Battery Charging Module	Digi-key	1	12.50	12.50
DC House Linear Actuator	Linear Actuator	Amazon	1	34.99	34.99
Adafruit Battery Pack	Battery	Mouser	1	24.50	24.50
ARM dual-core MCU	Micro-controller	Mouser	1	8.82	8.82
CONN RCPT USB2.0 MICRO B	Micro-USB Connector	Digi-key	2	0.77	1.54
CONN RCPT USB2.0 MINI B SMD	Mini-USB Connector	Digi-key	2	0.65	1.30
ECMF02-2AMX6	Common Mode Filter	Digi-key	2	0.54	1.08
IC REG LINEAR 3.3V	Linear Regulator	Digi-key	2	0.44	0.88
MOSFET 2N-CH 60V 0.305A	MOSFET	Digi-key	2	0.49	0.98
IC BATT CNTRL	Battery Management IC	Digi-key	2	1.14	2.28
IC REG BOOST ADJ 2A	Voltage Boost IC	Digi-key	2	1.24	2.48

Part Name	Part Description	Vendor	Quantity	Price per Unit	Total Price
RELAY TELECOM DPDT 2A 12V	DPDT Relay	Digi-key	2	3.71	7.42
TRANS NPN 25V 1.5A	Transistor	Digi-key	2	0.29	0.58
NRF24L01P-R7	2.4GHZ Transceiver	Digi-key	2	4.07	8.14
RES SMD 2K	2 k Resistor	Digi-key	2	0.48	0.96
RES 51K OHM	51 k Resistor	Digi-key	10	0.016	0.16
RES 13.3K OHM	13.3 k Resistor	Digi-key	10	0.019	0.19
CAP CER 22UF	22 uF Capacitor	Digi-key	2	0.13	0.26
IC REG LINEAR 5V	5V Linear Regulator	Digi-key	2	0.38	0.76
FIXED IND 10UH	10uH Inductor	Digi-key	2	0.25	0.50
FIXED IND 3.9NH	3.9nH Inductor	Digi-key	2	0.25	0.44
FIXED IND 8.2NH	8,2nH Inductor	Digi-key	2	0.25	0.44
FIXED IND 2.7NH	2.7nH Inductor	Digi-key	2	0.25	0.46
CONN HEADER SMD	Connector Header	Digi-key	2	0.51	1.02
ABM8-16.000MHZ	CRYSTAL 16MHZ	Digi-key	2	0.49	0.98
TOTAL					220.82

7.2 Requirements and Verifications Tables

7.2.1 Control Subsystem

Requirements	Verification
• The Bluetooth connection must have a reasonable range that must ignore a mobile device until it is within arms reach. Program for 1ft.	• Slowly approach the Control Subsystem from 1 meter away with an already paired Bluetooth device and record when the device automatically connects. • Slowly back away from the Control Subsystem and record when the connection disconnects. • Confirm that the record is 1ft. If not, double-check the limit range and consider what may interfere with the signal.
• Ensure the microcontroller is receiving 3.3-3.6 V from the battery.	• Use a multimeter to measure the voltage at the power input pin of the microcontroller.

7.2.2 Power Subsystem

Requirements	Verification
• Ensure the Power subsystem properly charges when the bike is not in motion.	• Use a multimeter to measure the voltage at the power input pin of the microcontroller when the bike is not in motion. It should read some amount of voltage.
• Ensure that the Power Subsystem does not discharge when the bike is in motion.	• Use a multimeter to measure the voltage at the power input of the microcontroller when the bike is in motion. It should not read any significant voltage.

7.2.3 Locking Subsystem

Requirements	Verification
• Ensure that the Boost Voltage Regulator boosts the voltage to 12 V for the linear actuator.	• Use a multimeter to measure the voltage at the power input pin of the linear actuator.
• Ensure the keypad works in both the scenario that the Bluetooth connection is active and inactive.	• Press any of the keys on the keypad when the Bluetooth connection is active. The status of the lock should change. • Enter a pattern into the keypad when the Bluetooth connection is not active. If the combo matches the set pattern, the lock status should change. Otherwise, the lock status should not change.
• Ensure the linear actuator works properly as a physical locking mechanism.	• Attempt to roll the bike when the linear actuator is extended. It should not be able to move.

7.2.4 Alarm Subsystem

Requirements	Verification
• Ensure that the Alarm Subsystem can ignore small and periodic motions that are likely due to wind, bumping, etc.	• Simulate by nudging the bike with little force.
• Ensure that the Alarm Subsystem can recognize substantial and continuous vibrations.	• Simulate by picking up and carrying the bike away. • Simulate by fidgeting with the components and frame of the bike.