Table No. : 4

Section: A

Team Members:

Anubhav Singh (210163)

Anushree Chaudhary (210176)

Ankit Khandelwal (210149)

EE381A- EC Lab Project Report

Knock-Detecting Security Mechanism

Q1: What problem are you trying to solve, and why is it important/interesting?
Ans. Our project involves making an electronic security lock system that can unlock a door using a knock-detecting mechanism and we can also see it's working on the mobile app connected via Bluetooth. Our project adds a layer of security by requiring a specific action or signal to unlock the door, which can help prevent unauthorized individuals from gaining entry through traditional methods like picking locks or using stolen keys. It is programmed to grant access to specific individuals or groups and can be easily reprogrammed when required. For example, our project can be used perfectly at secret labs by making the pattern harder and harder. One may customize the number of knocks, along with the time intervals between each knock. It can also provide an extra layer of security at required places.

Q2: What are the existing solutions? Describe a few of them and list any shortcomings in them. Is your solution approach unique in some way?
Ans. Existing solutions for lock mechanisms include traditional mechanical locks, electronic locks, and smart locks. Here are a few examples along with their shortcomings:

1. Mechanical Locks: The keys must be carried physically and may be misplaced. Also, it needs multiple keys for multiple users.
2. Deadbolt Locks: These provide good security but can be susceptible to picking and lock bumping techniques.
3. Knob Locks: They are less secure than deadbolts and can be easily forced open with enough pressure or leverage.
4. Electronic Locks: Keypad locks: They require a PIN code to unlock which is vulnerable to code guessing or shoulder surfing.
5. Biometric Locks: Use fingerprints, iris scans, or other biometric data for access. They may have reliability issues with recognition and can be fooled with replicas of biometric data.
6. Smart Locks: Bluetooth/Wi-Fi Enabled Locks: They allow remote unlocking via smartphone apps which are vulnerable to hacking if not properly secured and may have connectivity issues.
7. Voice-Activated Locks: They unlock using voice commands. It may be activated unintentionally or by someone mimicking the owner's voice.

The knock-detecting mechanism is unique because it relies on a specific pattern of knocks or taps to grant access, adding an extra layer of security. Its approach is unique in that it doesn't require traditional keys, codes, or biometric data, making it less susceptible to hacking or replication.

Q3: What resources do you require to complete the project? Give a breakdown of the tasks that you need to accomplish week by week to complete the project.
Ans. Resources required:

1. Arduino NANO
2. Piezoelectric Sensor
3. Relay module - JQC-3FF-S-Z
4. Solenoid lock
5. Bluetooth module – HC-05
6. 12V DC power supply
7. Jumper wires
8. Breadboard
9. USB 2.0 Cable Type A/B
10. Serial Bluetooth Terminal (Application on Mobile)
11. LED
12. Resistors (10kΩ and 220Ω)

Week 1: Getting familiar with all the components, such as the Arduino NANO, piezoelectric sensor, and solenoid lock.
Week 2: Assembling all the components and making the entire circuit.
Week 3: Making the application, integrating it with the hardware, along with its testing.

Project Objective:

The goal of this project is to create a solenoid lock that can notify the owner's mobile device and open with a specific knocking pattern that is picked up by a piezoelectric sensor. The message informs the recipient that the door has been unlocked, and if there has been a failed attempt at unlocking the security mechanism. This could improve room security by offering a practical and safe method of access control.

Resources Used:

1. Arduino NANO
2. Piezoelectric Sensor
3. Relay module – JQC-3FF-S-Z
4. Solenoid lock
5. Bluetooth module – HC-05
6. 12V DC power supply
7. Jumper wires
8. Breadboard
9. USB 2.0 Cable Type A/B

10. Serial Bluetooth Terminal (Application on Mobile)
11. LED
12. Resistors (10kΩ and 220Ω)

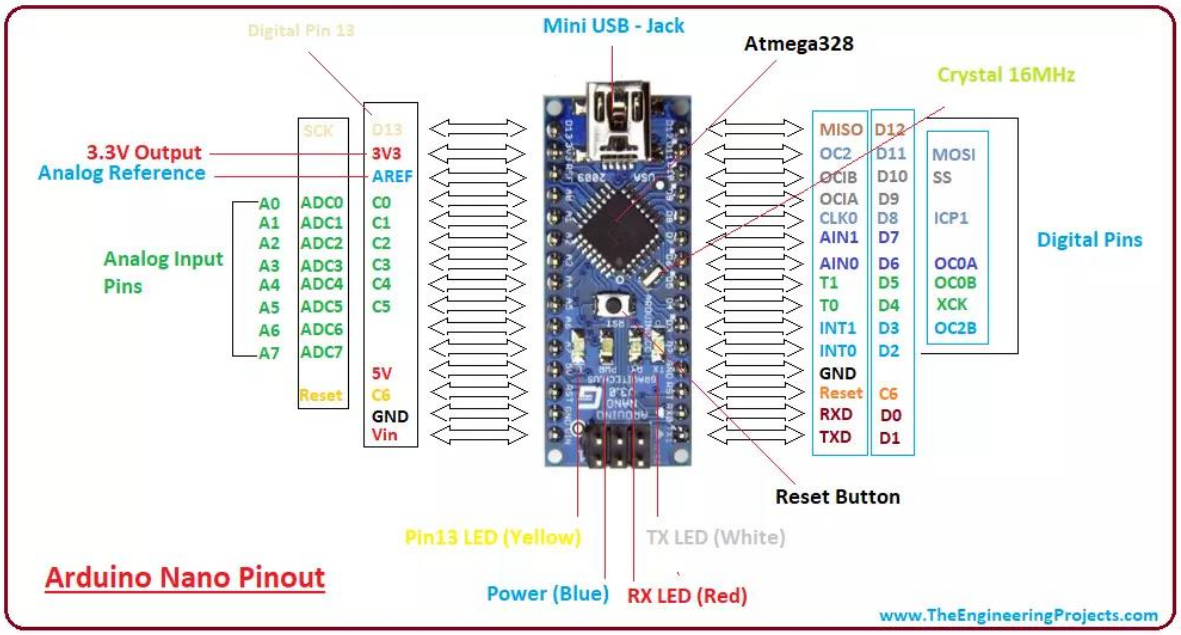
Description of components used:

1. Arduino NANO:

The Arduino NANO is a compact and versatile microcontroller board, suitable for a wide range of electronic projects. It is based on the ATmega328P microcontroller, just like the Arduino UNO, and offers similar functionality in a smaller form factor. Some of its key specifications are:

a) Microcontroller: ATmega328P
b) Operating Voltage: 5V
c) Input Voltage: 7-12V
d) Digital I/O pins: 14 (of which 6 provide PWM output)
e) Analog Input Pins: 8
f) Flash Memory: 32KB (of which 2KB is used by bootloader)
g) SRAM: 2KB
h) Clock Speed: 16MHz
i) USB Interface: Mini-B

The Arduino NANO shares many features with the Arduino UNO but is smaller in size, making it suitable for projects with space constraints or where a more compact design is desired. It is widely used by students and professionals alike for its versatility and ease of use.



2. Piezoelectric Sensor

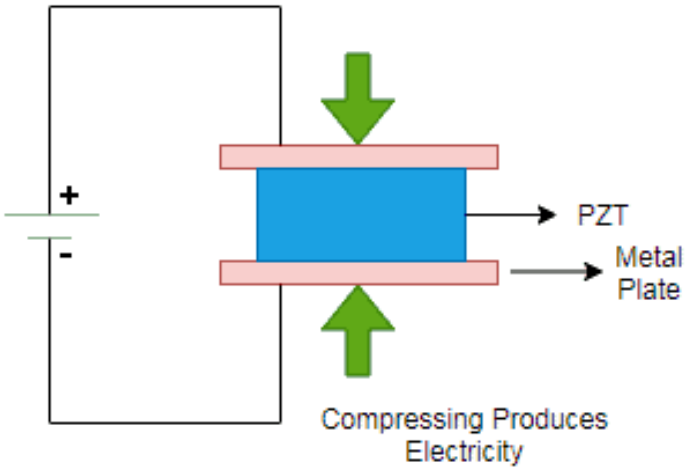
The piezoelectric sensor is a type of sensor that utilizes the piezoelectric effect to generate an electrical charge when mechanical stress or vibration is applied to it. These sensors are commonly used in various applications for detecting pressure,

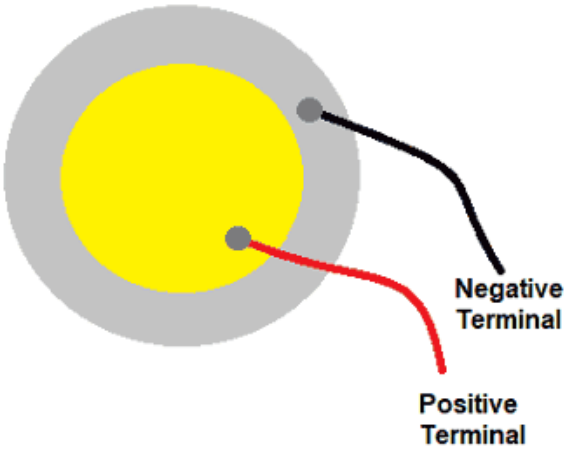
force, acceleration, and vibration. Some key specifications of a typical piezoelectric sensor:

a) Operating Voltage: Typically, piezoelectric sensors operate within a wide range of voltages, depending on the specific model and application requirements.
b) Sensitivity: Piezoelectric sensors have high sensitivity, capable of detecting even minute changes in mechanical stress or vibration.
c) Frequency Response: The frequency response of piezoelectric sensors varies based on the specific model and application, typically ranging from a few hertz to several kilohertz.
d) Output Signal: Piezoelectric sensors generate an electrical charge proportional to the applied mechanical stress or vibration, producing an output voltage or current signal.
e) Dynamic Range: Piezoelectric sensors have a wide dynamic range, enabling them to detect both small and large magnitudes of mechanical stress or vibration.
f) Operating Temperature Range: Piezoelectric sensors are designed to operate within specific temperature ranges, ensuring reliable performance in various environmental conditions.
g) Size and Dimensions: Piezoelectric sensors come in various sizes and shapes, with dimensions tailored to specific applications.
h) Material: Piezoelectric sensors are typically made of piezoelectric materials such as quartz, PZT (lead zirconate titanate), or PVDF (polyvinylidene fluoride).
i) Application: Piezoelectric sensors are commonly used in applications such as vibration monitoring, impact detection, acoustic measurements, force sensing, and pressure sensing.

Overall, piezoelectric sensors offer high sensitivity, fast response times, and robustness, making them suitable for a wide range of industrial, automotive, aerospace, and biomedical applications.

In our project, the piezoelectric sensor mapped the 10-bit output to voltage levels 0V to 5V, and we set the threshold accordingly so that it is only crossed when we apply pressure and is not affected by surroundings.





3. Relay Module:

The JQC-3FF-S-Z is a commonly used relay module that is designed to switch high-voltage loads using low-voltage control signals. It is widely used in various applications such as industrial control, home automation, and robotics.

Some of the key specifications of the JQC-3FF-S-Z relay module are:

i) Number of Relays: 1
ii) Input Voltage: 5V DC
iii) Maximum Load Voltage: 250V AC or 30V DC
iv) Maximum Load Current: 10A
v) Operating Temperature: -40°C to 70°C
vi) Contact Form: SPDT (Single Pole Double Throw)
vii) Relay Type: Electromechanical
viii) Control Interface: Digital Input
ix) Mounting Type: PCB

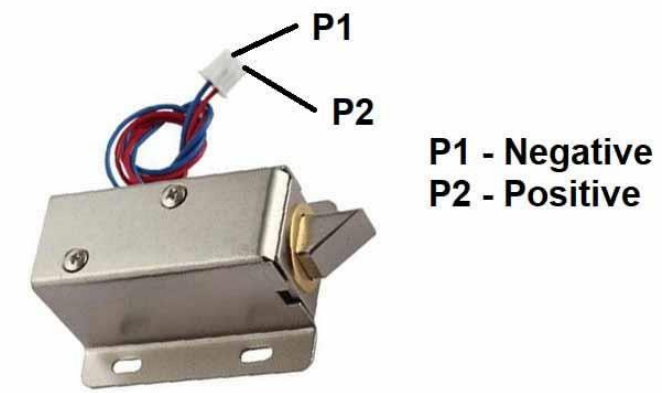
4. Solenoid Lock

A 12V DC solenoid lock is an electromechanical device that is commonly used to secure doors, cabinets, and other access control points. When an electric current is

applied to the solenoid, a magnetic field is generated, which pulls a metal plunger into the coil, thus locking the mechanism.

Here are some specifications of a typical 12V DC solenoid lock:

i) Operating Voltage: 12V DC
ii) Operating Current: 400mA to 500mA
iii) Operating Temperature: -20°C to 55°C
iv) Material: Stainless Steel
v) Holding Force: 600lbs (approx. 272kg)
vi) Lock Type: Fail-Secure
vii) Lock Mechanism: Solenoid
viii) Mounting Type: Surface Mount
ix) Dimensions: 150mm x 28mm x 28mm

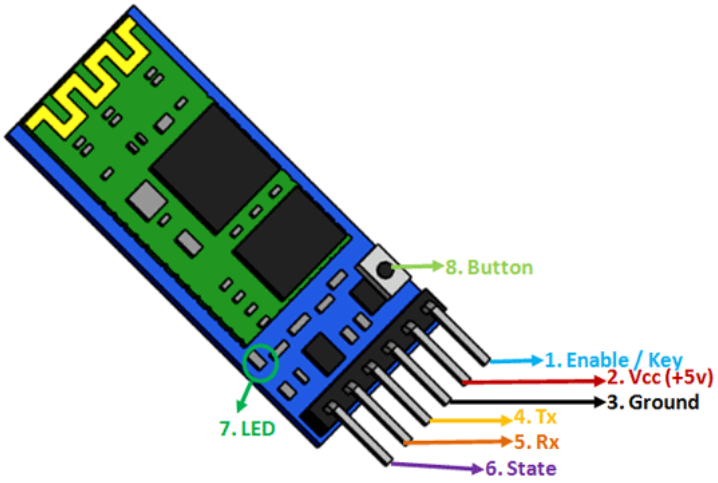


5. Bluetooth module – HC-05

The HC-05 Bluetooth module is a popular module that allows wireless communication between devices over Bluetooth. It uses the Bluetooth 2.0+EDR (Enhanced Data Rate) protocol and can be configured as either a master or a slave device.

Here are some specifications of the HC-05 module:

i) Bluetooth Version: Bluetooth 2.0+EDR
ii) Frequency Band: 2.4 GHz ISM band
iii) Modulation: GFSK (Gaussian Frequency Shift Keying)
iv) Transmit Power: Class 2, up to 4dBm
v) Sensitivity: -84dBm at 0.1% BER
vi) Range: Up to 10 meters (Class 2)
vii) Operating Voltage: 3.3V DC to 6V DC
viii) Current Consumption: <30mA (at 3.3V DC)
ix) Interface: UART (Universal Asynchronous Receiver/Transmitter)
x) Baud Rate: Default 9600 baud, configurable up to 1382400 baud
xi) Dimensions: 28mm x 15mm x 2.35mm



Implementation

A. Connections:

i) Bluetooth module:
Vcc → +5V of Arduino NANO
Gnd→ Gnd of Arduino NANO
Tx → D10 of Arduino NANO
Rx → D11 of Arduino NANO

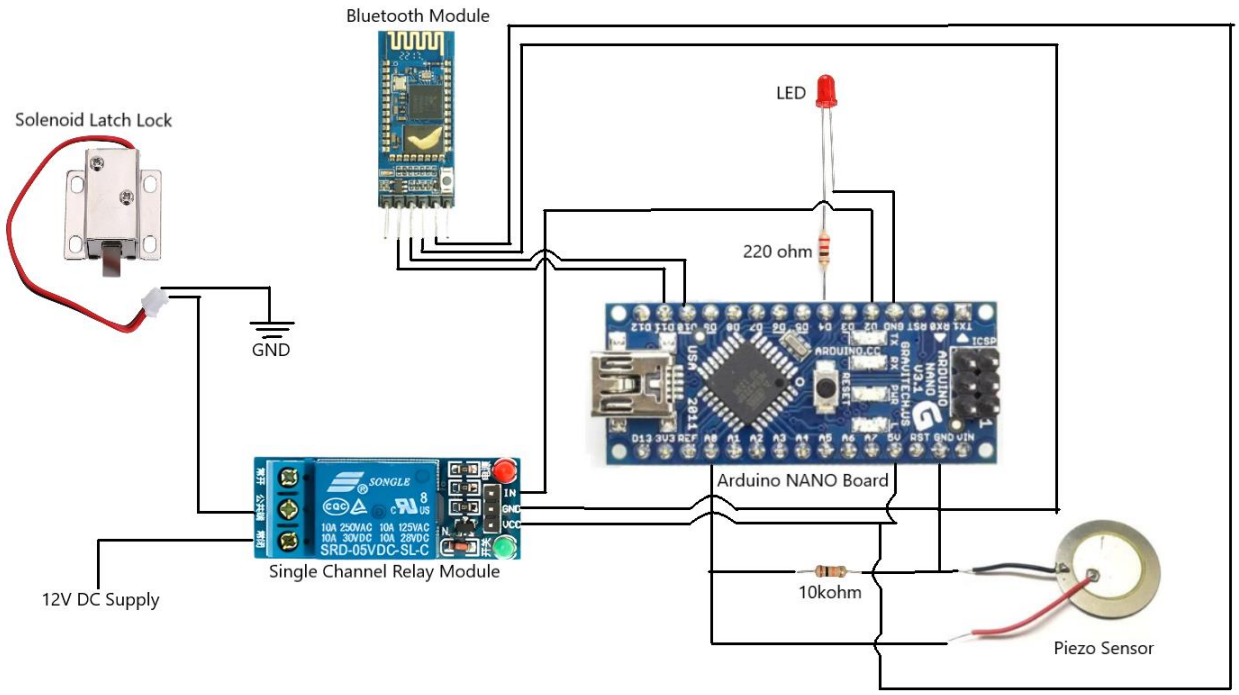
ii) Solenoid Lock:
Positive P1 → COM of Relay module
Negative P2 → Negative/Ground of 12V DC Supply

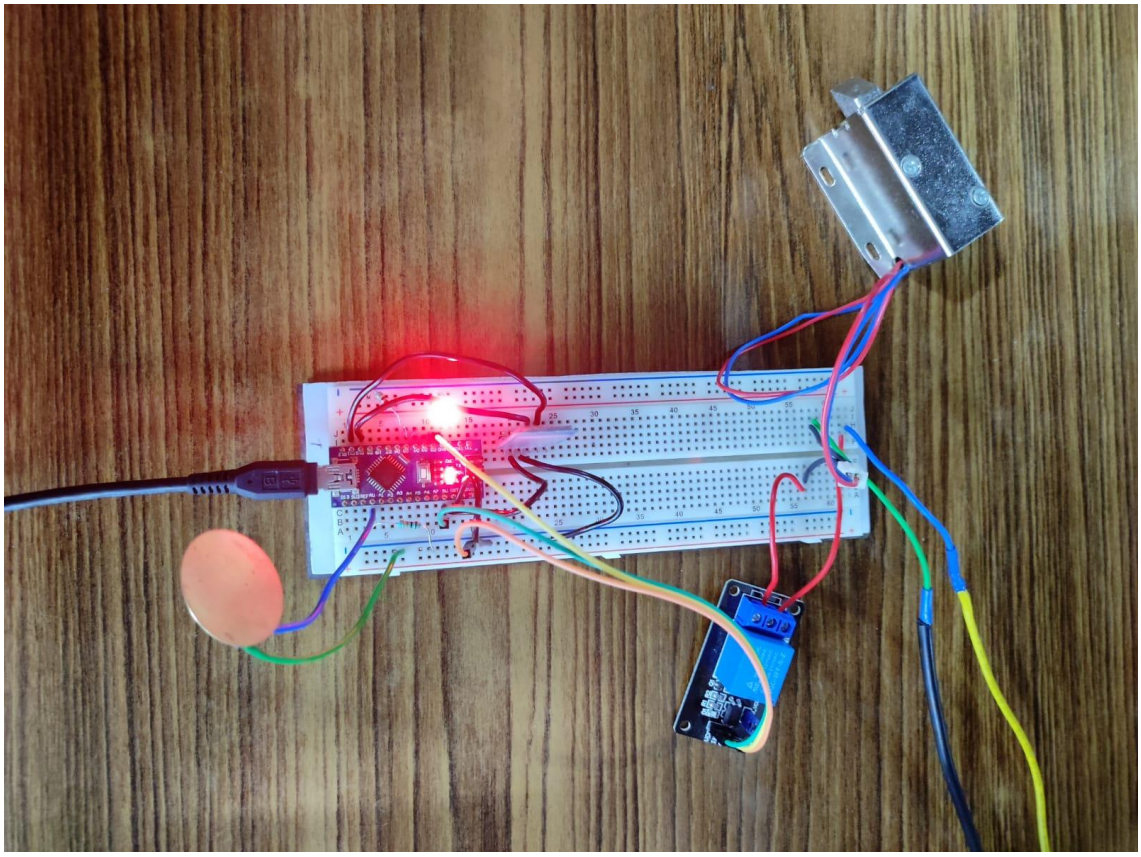
iii) Relay Module:
Vcc → +5V of Arduino NANO
Gnd → Gnd of Arduino NANO
In1 → D2 of Arduino NANO
COM → Positive of Solenoid lock
NO (Normally open) → Positive terminal of 12V DC Supply

iv) Piezoelectric Sensor:
Positive Terminal → A0
Negative Terminal → GND of Arduino NANO

v) Arduino NANO
USB B port →USB A port of Laptop

B. Final Circuit:





C. Programming the setup:

The final code that we used is:

```
#include <SoftwareSerial.h>// import the serial library

SoftwareSerial BT(10, 11); // TX, RX
// Pin definitions
const int knockSensor = A0; // Piezo sensor on pin 0.
const int LED = 4;  // Status LED

// Tuning constants. Could be made vars and hoooked to potentiometers for soft configuration, etc.
const int threshold = 10;  // Minimum signal from the piezo to register as a knock
const int rejectValue = 25; // If an individual knock is off by this percentage of a knock we don't unlock..
const int averageRejectValue = 15;  // If the average timing of the knocks is off by this percent we don't unlock.
const int knockFadeTime = 150;  // milliseconds we allow a knock to fade before we listen for another one. (Debounce timer.)
const int maximumKnocks = 10; // Maximum number of knocks to listen for.
const int knockComplete = 1200;  // Longest time to wait for a knock before we assume that it's finished.

#define RELAY_PIN  2 // The Arduino Nano pin connected to the IN pin of relay


// Variables.
int secretCode [maximumKnocks] = {100, 100, 0, 0, 0, 0, 0, 0, 0, 0}; // Secret Lock Combination
int knockReadings [maximumKnocks]; // When someone knocks this array fills with delays between knocks.
int knockSensorValue = 0; // Last reading of the knock sensor.

void setup() {
// put your setup code here, to run once:
  BT.begin(9600);   // Uncomment the Serial.bla lines for debugging.
  BT.println("Bluetooth On please wait....");   // but feel free tomment them out after it's working right.
  pinMode(LED, OUTPUT);
  pinMode(knockSensor, INPUT);

  // initialize digital pin 2 as an output.
  pinMode(RELAY_PIN, OUTPUT);

  digitalWrite(LED, HIGH);   // Red LED on, everything is go..
  // delay(1000);
  // digitalWrite(LED, LOW);
}

void loop() {
  // delay(1000);
  // digitalWrite(LED, LOW);
  digitalWrite(RELAY_PIN, HIGH);
// Listen for any knock at all.
  knockSensorValue = analogRead(knockSensor);

  if (knockSensorValue >=threshold){
    listenToSecretKnock();
  }
}

// Records the timing of knocks.
void listenToSecretKnock(){
  BT.println("knock starting");

  int i = 0;
  // First lets reset the listening array.
  for (i=0;i<maximumKnocks;i++){
    knockReadings[i]=0;
  }

  int currentKnockNumber=0;   // Incrementer for the array.
  int startTime=millis();   // Reference for when this knock started.
  int now=millis();

  digitalWrite(LED, LOW);   // we blink the LED for a bit as a visual indicator of the knock.
  delay(knockFadeTime);   // wait for this peak to fade before we listen to the next one.
```

```
  digitalWrite(LED, HIGH);

  do {
    //listen for the next knock or wait for it to timeout.
    knockSensorValue = analogRead(knockSensor);
    if (knockSensorValue >=threshold){    //got another knock...
    //record the delay time.
      BT.println("knock.");
      now=millis();
      knockReadings[currentKnockNumber] = now-startTime;
      currentKnockNumber ++;    //increment the counter
      startTime=now;
      // and reset our timer for the next knock
      digitalWrite(LED, LOW);
      delay(knockFadeTime);   // again, a little delay to let the knock decay.
      digitalWrite(LED, HIGH);

    }

    now=millis();

    //did we timeout or run out of knocks?
  } while ((now-startTime < knockComplete) && (currentKnockNumber <maximumKnocks));

  //we've got our knock recorded, lets see if it's valid
          // only if we're not in progrmaing mode.
    if (validateKnock() == true){
      triggerDoorUnlock();
    } else {
      BT.println("Secret knock failed.");
      digitalWrite(LED, LOW);   // We didn't unlock, so blink the red LED as visual feedback.
      digitalWrite(LED, HIGH);
    }
```

```

}

void triggerDoorUnlock(){

  int i=0;
// Blink the green LED a few times for more visual feedback.
  for (i=0; i < 5; i++) {
    digitalWrite(LED, LOW);
    delay(100);
    digitalWrite(LED, HIGH);
    delay(100);
  }
  BT.println("Door unlocked!");
  digitalWrite(RELAY_PIN, LOW); // unlock the door
  delay(5000);
  BT.println("Door locked!");
  digitalWrite(RELAY_PIN, HIGH);  // lock the door
  delay(5000);
}

// Sees if our knock matches the secret.
// returns true if it's a good knock, false if it's not.
// todo: break it into smaller functions for readability.
boolean validateKnock(){
  int i=0;

  // simplest check first: Did we get the right number of knocks?
  int currentKnockCount = 0;
  int secretKnockCount = 0;
  int maxKnockInterval = 0;    // We use this later to normalize the times.

  for (i=0;i<maximumKnocks;i++){
```

```
    if (knockReadings[i] > 0) {
      currentKnockCount++;
    }
    if (secretCode[i] > 0) {    //todo: precalculate this.
      secretKnockCount++;
    }

    if (knockReadings[i] > maxKnockInterval) {  // collect normalization data while we're looping.
      maxKnockInterval = knockReadings[i];
    }
  }

  // If we're recording a new knock, save the info and get out of here.

  if (currentKnockCount != secretKnockCount) {
    return false;
  }

  /* Now we compare the relative intervals of our knocks, not the absolute time between them.
    (ie: if you do the same pattern slow or fast it should still open the door.)
    This makes it less picky, which while making it less secure can also make it
    less of a pain to use if you're tempo is a little slow or fast.
  */
  int totaltimeDifferences=0;
  int timeDiff=0;
  for (i=0;i<maximumKnocks;i++){ // Normalize the times
    knockReadings[i]= map(knockReadings[i],0, maxKnockInterval, 0, 100);
    timeDiff = abs(knockReadings[i]-secretCode[i]);
    BT.println(knockReadings[i]);
    if (timeDiff > rejectValue) { // Individual value too far out of whack
      return false;
    }
    totaltimeDifferences += timeDiff;
  }
  // It can also fail if the whole thing is too inaccurate.
  if (totaltimeDifferences/secretKnockCount>averageRejectValue){
    return false;
  }

  return true;

}
```

This code compares the unlocking knock sequence we have set with the input knock sequence that has been given by adding pressure to the piezoelectric sensor. In doing so, to ensure that the pattern and not exact time intervals are compared, we have normalized the intervals between knocks before comparing. There are three stages of verifying if the knock is correct: the number of knocks is the same, the time intervals between any two knocks are within the 25% tolerance range, and the average time difference tolerance is in the 15% tolerance range.

- Explanation:
  (1) We have used “<SoftwareSerial.h>”, this library has functions that can make any digital output as Rx and Tx.
  (2) D11 and D10 are set Rx and Tx respectively, for Bluetooth module.
  (3) D2 is set as Output, which becomes the input for the relay.
  (4) The function of the code is to match the received knock sequence with the saved one. When the sequence is matched successfully, then the Output (D2) is made high, accompanied by blinking of the LED 5 times. We have added a delay of 5s using delay (5000) and then made Output (D2) low.
  (5) When the Output (D2) is made high, the relay input gets high and thus the relay switch gets closed, as we are using NO mode (normally open) of the relay, making the solenoid lock circuit close, and hence opening the lock.

(6) When the Output (D2) is again made low, the relay input gets low and thus the relay switch gets opened, making the solenoid lock circuit open and hence closing the lock again.
(7) After the sequence is matched, Arduino gives the command to the Bluetooth module to print the door is unlocked message on the Terminal of Serial Bluetooth application.

Testing and Results:

I. After the setup is ready, turn ON the 12V power supply.
II. We have powered the Arduino NANO board using the laptop itself.
III. Perform the knock sequence on the piezo sensor, if the sequence matches, the lock opens up, remains open for 5 sec and then closes again.
IV. The terminal of the Serial Bluetooth application will have a message printed “The door is unlocked” at the time of unlocking.
V. A 5-second delay does not mean that the door must be again closed within this 5-second duration. Whenever the door is closed, due to a special build-up of the Solenoid lock which when not powered with 12V, the force of a closing door pushes the lock inside and allows it to pass through it, and then the solenoid lock comes out and locks the closed door. Now to open the door, we need to open the lock, which happens when the knock pattern is matched again.
VI. When a wrong knock pattern is detected, the solenoid lock remains unaffected and the door remains locked, with the Bluetooth module sending a message “Secret knock failed”.

Conclusion:

In conclusion, the project of making a solenoid lock that can be opened by doing a specific knock sequence and sends a notification to the owner's mobile phone saying that the door has been unlocked (or if the unlocking has failed) can be achieved using the following components:

a. Arduino Nano - to control the overall system
b. Piezoelectric Sensor - to input and send the knock pattern
c. Relay module JQC-3FF-S-Z - to control the solenoid lock
d. 12V DC solenoid lock - to lock and unlock the door
e. Bluetooth module HC-05 - to send notifications to the owner's mobile phone.

By integrating these components and programming the Arduino to control the system, we can create a secure and convenient access control solution for our project. The user can knock on the surface of the door which will stimulate the piezoelectric sensor, which will match the entered sequence with the registered one and send a signal to the Arduino to unlock the solenoid lock via the relay module. The Bluetooth module can then send a notification to the owner's mobile phone indicating that the door has been unlocked. This project can be useful for securing hostel rooms and any other secretive lock-requiring applications.

A video is linked below showing how the mechanism would work for a correct knock sequence and how it would fail if a different knock pattern is detected:

https://drive.google.com/file/d/1TfmBAf5rqL2_IFmlTPHUJOItlPJ9ZuEe/view?usp=sharing