Name: Roll Number:
Vinit Doke 190260018
Mihir Agre 190260030
Vignesh Tongaria 190260046

Pattern Unlock using Ultrasound Sensors

Problem statement

We will use an Arduino board and a set of Ultrasonic Distance Sensors to map a small 2D region of space to a 3×3 grid. To achieve it, we will test 2 configurations of sensor placements: sensors in a same line / when they are perpendicular. Next, we will use the coordinates (discrete) obtained from detecting an object (or Hand) from the previous step to make a 3×3 Pattern Lock.

Team

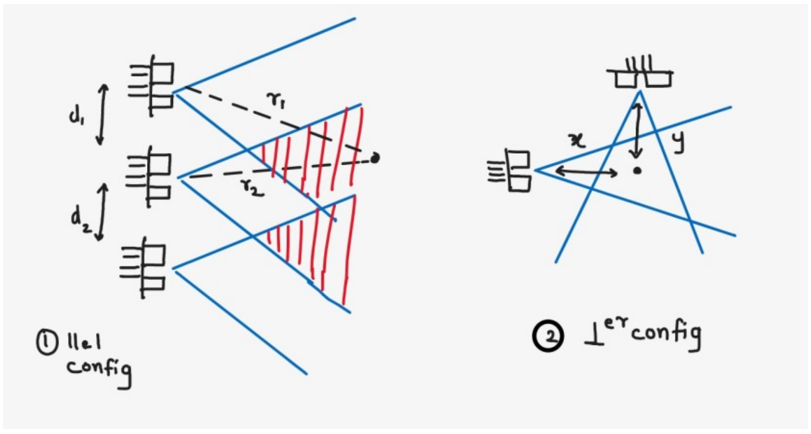
Vinit Doke
Mihir Agre
Vignesh Tongaria

Components Required

- Arduino UNO
- Ultrasonic Distance Sensors (2)
- 3×3 LED Grid
- 74HC595 Shift Register

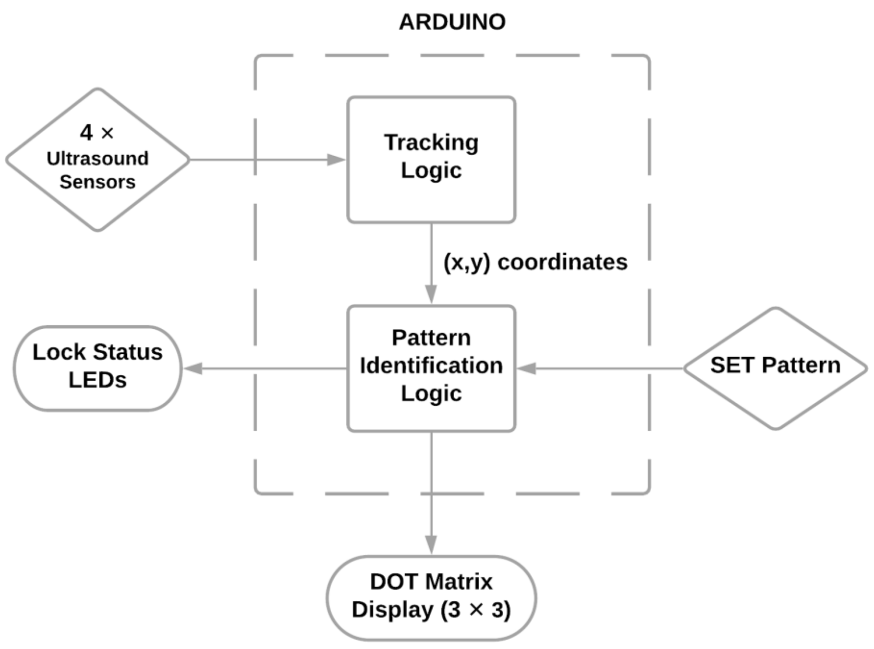
Description

In order to track an object in 2D Space, we will be using multiple ultrasonic distance sensors in one of the following configurations after due experimentation and weighing their pros and cons individually. The configurations are as follows:
1. Array of ultrasound sensors placed parallel to each other and using triangulation to calculate the position of the detected object.
2. Ultrasound Distance Sensors placed perpendicular to each other, which individually output the x and y coordinates.

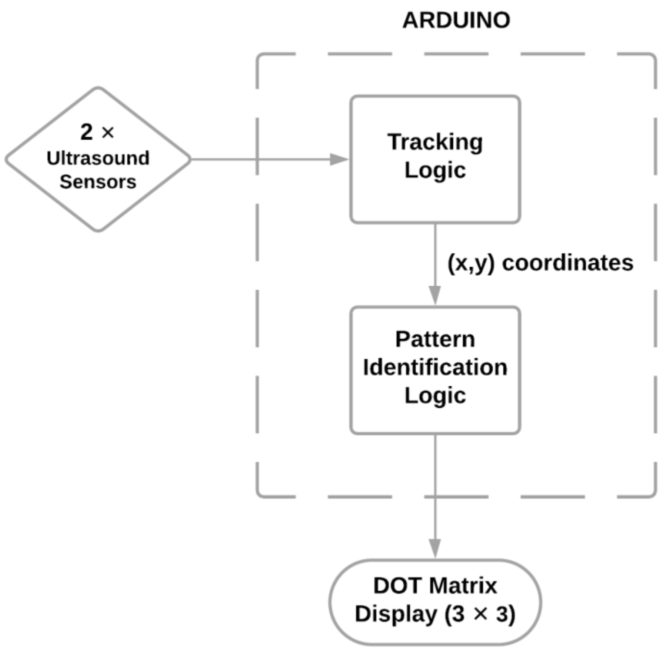


Name:	Roll Number:
Vinit Doke	190260018
Mihir Agre	190260030
Vignesh Tongaria	190260046

Block diagram



UPDATED :



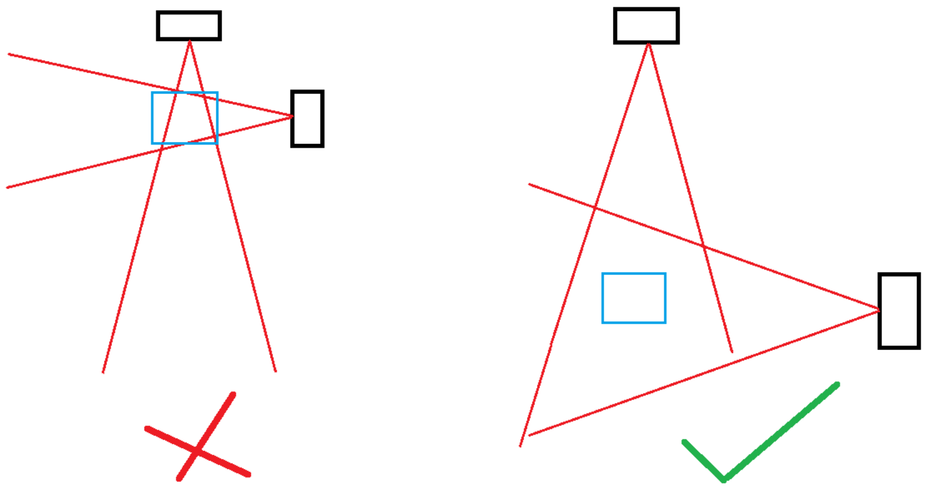
Status is Indicated by blinking different patterns on the display and password can be reset in-code by reprogramming the Arduino.

Name: Roll Number:
Vinit Doke 190260018
Mihir Agre 190260030
Vignesh Tongaria 190260046

Design details

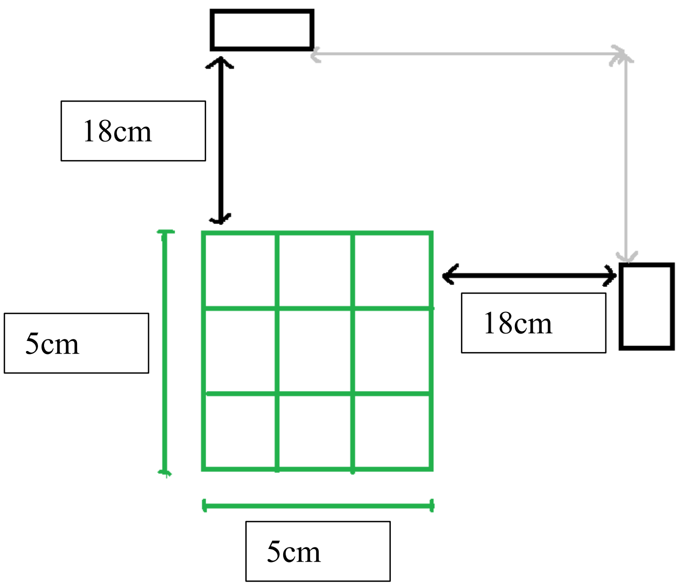
1. Calibration

As indicated in the block diagram, the ultrasonic sensors are used to get readings of position of an object in their common field of view. We know that sensors act only in a certain area so it is important to first find a good region where our readings wouldn't bounce randomly.



Black are sensors and the region inside red lines indicate their area of actions.

Let's call this process **calibration**. To calibrate the spatial configuration of our sensors and region of interest we need to know the geometry of cone in which sensor acts, which we find by sweeping an object of interest across a single dimension (pertaining to one of the sensors) at a time. Now based on this cone we find the region of interest by hit and trials depending on the thresholds that are adequate.



Name:	Roll Number:
Vinit Doke	190260018
Mihir Agre	190260030
Vignesh Tongaria	190260046

2. Pattern Storage and Display

Next after a region is obtained and divided into 3*3 grid, we proceed to pattern recognition logic. This is basically storing the lighted LED in arrays, i.e., remembering which cell of the grid was accessed.

At any given point, we individually fire the ultrasound sensors and obtain the average of the distance readings over 100 instances. This gives us a fairly smooth distance-time graph as opposed to a noisy-one sampled for lesser instances. The individual distances are then thresholded according to the measurements done in the Calibration part, thus, we finally obtain the block of the grid that was accessed at the moment.

This recognised pattern is then compared with the stored password. Doing all this blindly is no fun so, we have a LED display to show the pattern and the successful/unsuccessful logins.

This was done in 2 ways:

A. Using 74HC595 IC :

This is a serial-input shift register with 8-bits of storage, each corresponding to one of the LEDs in the 3 x 3 Matrix and the last LED is controlled directly by a GPIO pin on the Arduino. The advantage of using the IC was that we had extra pins on the Arduino that could be repurposed for newer inputs like buttons, as well as adding in more ultrasound sensors to increase the area of detection.

B. Individual LEDs Controlled using individual pins on the Arduino UNO.

3. Pattern Verification Method

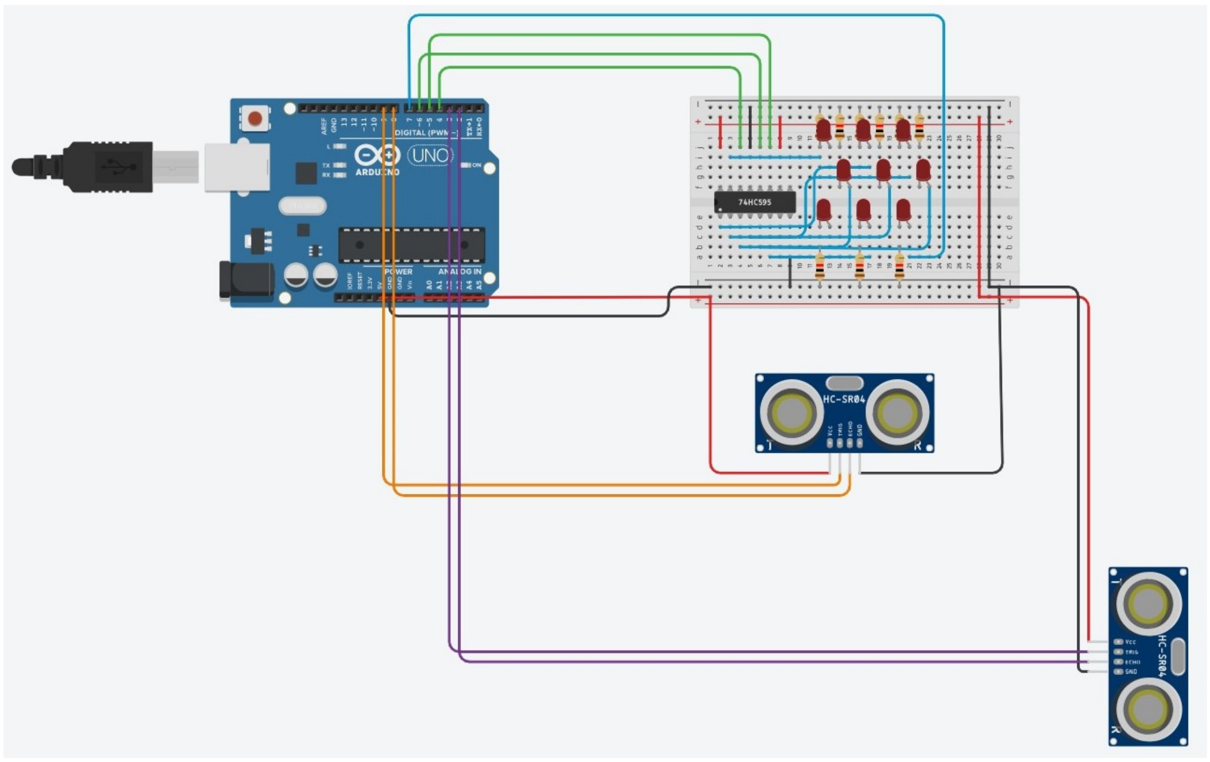
Given the very noisy nature of the detections we obtained, we simplified the conditions for a successful "Unlock". As long as the generated pattern is visually similar to the stored password, it does not matter what path was taken to generate that pattern. This is in contrast with what generally happens in say, Android smartphone locks, which require the pattern to be drawn in exact same manner as the password.

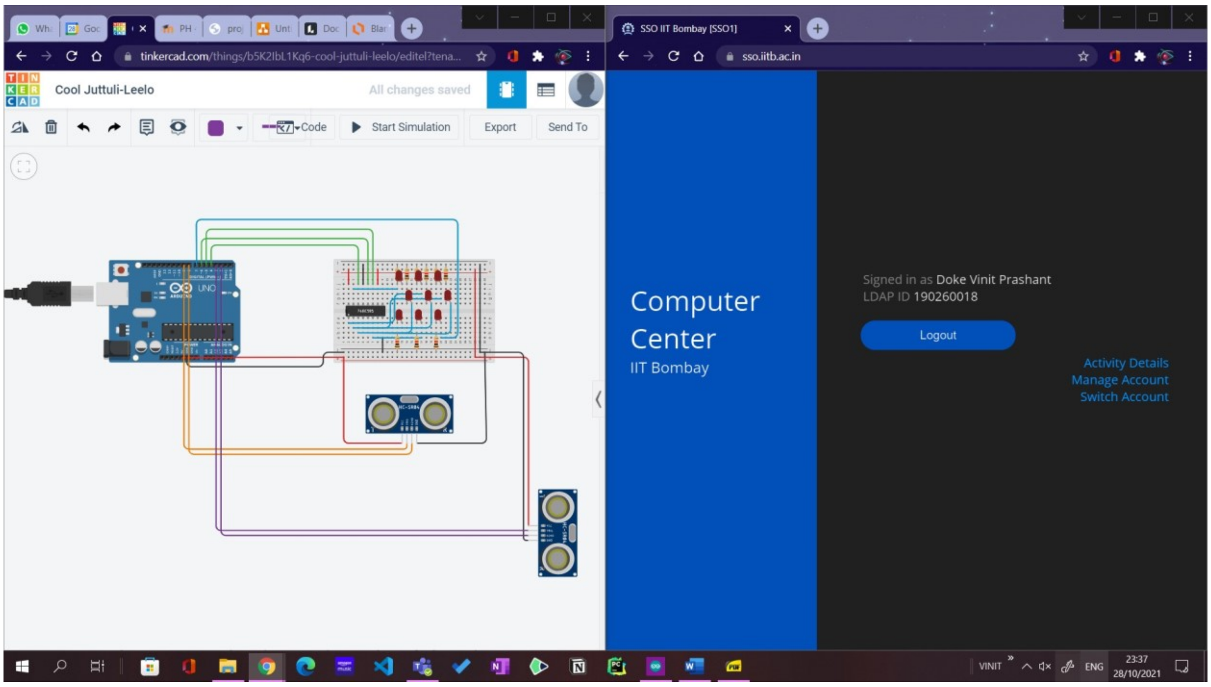
For instance, if the set password was a straight vertical line, moving the object from up to down, or down to up & or finally, from center to top (bottom) to bottom (top) would be acceptable for an Unlock condition. This obviously makes the lock less secure but it allows for better operational ease, accounting for the noisy detections made by the ultrasound sensors.

Name:	Roll Number:
Vinit Doke	190260018
Mihir Agre	190260030
Vignesh Tongaria	190260046

Circuit diagram and image

Circuit Diagram with 74HC595 IC :





Computer Center
IIT Bombay

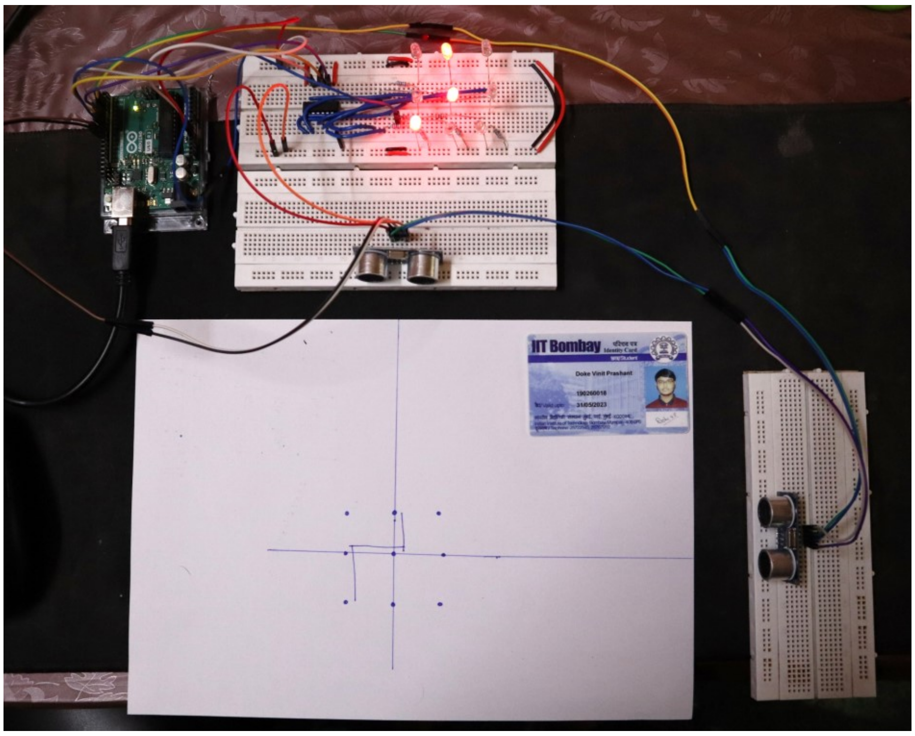
Signed in as Doke Vinit Prashant
LDAP ID 190260018

Logout

Activity Details
Manage Account
Switch Account

Name:	Roll Number:
Vinit Doke	190260018
Mihir Agre	190260030
Vignesh Tongaria	190260046

Implementation Image : (Vinit Doke, 190260018)



<u>Demonstration Link:</u>

1. GDrive:
https://drive.google.com/drive/folders/1XKPaD3ARXW_q0r8hr54P34GfqiNOwEWA?usp=sharing

2. OneDrive:
https://iitbacin-my.sharepoint.com/:f:/g/personal/190260018_iitb_ac_in/EmBuQlF-3kZFru6eAqXzyLgBvlQRD_CUIPNYv0t9qDDprQ?e=5TVp8R

Name: Roll Number:
Vinit Doke 190260018
Mihir Agre 190260030
Vignesh Tongaria 190260046

Bill of materials

- 2 Ultrasound Distance Sensors (HC-SR04)
- 9 LEDs
- IC : 74HC595
- Cables, Resistors

Status and conclusion

In conclusion, we have successfully made and demonstrated a setup to track an object in a small 2D space using Ultrasound Distance Sensors, and applied the same principle to implement a pattern lock on a discrete 3 x 3 grid. The detection resolution varies between 1cm – 2cm and depends highly on the quality of the ultrasound sensor being used, precision of the calibration done during the setup stage and size of the object being tracked.

A particularly challenging problem we encountered during the implementation of the project is imprecise nature of the distance being recorded by the sensors, which at times would yield wildly different values for the same position of the object keeping all other factors constant.

Finally, we learnt a lot about the importance of correctly processing the time series data obtained from the sensors, adding buffers and eliminating sporadic fluctuations from the actual inference of the position of the detected object.

Name: Roll Number:
Vinit Doke 190260018
Mihir Agre 190260030
Vignesh Tongaria 190260046

Source codes

```
int dataPin = 4;
int clockPin = 5;
int latchPin = 6;
int lastLED = 7;

int xTrig = 3;
int xEcho = 2;

float X;
float Y;
int Xdiscrete;
int Ydiscrete;


int yTrig = 9;
int yEcho = 8;
long duration;

int wrong[9] = {1,0,1,
                0,1,0,
                1,0,1};
int frame1[9] = {1,1,1,
                 0,0,0,
                 0,0,0};
int frame2[9] = {0,0,0,
                1,1,1,
                0,0,0};
int frame3[9] = {0,0,0,
                 0,0,0,
                 1,1,1};

int password[9] = {0,1,0,
                   1,1,0,
                   1,0,0};

int currentStatus[9] = {0,0,0,
                        0,0,0,
                        0,0,0};

int xCoords[100];
int yCoords[100];



void setup() {

  Serial.begin(9200);
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(lastLED, OUTPUT);

  pinMode(xTrig, OUTPUT);
  pinMode(xEcho, INPUT);
  pinMode(yTrig, OUTPUT);
  pinMode(yEcho, INPUT);

}

void clearLEDs(){
  digitalWrite(latchPin, LOW);

  for (int i = 0; i < 8; i++){
    digitalWrite(clockPin, LOW);
    digitalWrite(dataPin, LOW);
    digitalWrite(clockPin, HIGH);
    }

   digitalWrite(lastLED, LOW);
   digitalWrite(latchPin, HIGH);

  }

void displayPattern(int arr[]){

  clearLEDs();
  digitalWrite(latchPin, LOW);

  for (int i = 7; i >= 0; i--){
    digitalWrite(clockPin, LOW);
    if(arr[i] == 0){
      digitalWrite(dataPin, LOW);
      digitalWrite(clockPin, HIGH);
      }
    else{
      digitalWrite(dataPin, HIGH);
      digitalWrite(clockPin, HIGH);
      }
    }
    if (arr[8] == 0){
    digitalWrite(lastLED, LOW);
    }
    else{
      digitalWrite(lastLED, HIGH);
      }
   digitalWrite(latchPin, HIGH);

  }


void wrongPattern(){
  for (int i = 0; i<=4; i++){
  displayPattern(wrong);
  delay(500);
  clearLEDs();
  delay(500);
  }
  }

void correctPattern(){
  displayPattern(frame1);
  delay(100);
  displayPattern(frame2);
  delay(100);
  displayPattern(frame3);
  delay(100);
  clearLEDs();
  }

double avg(int h[100]){

  double sum=0.0;

  for (int i=0; i<100; i++){
    sum = sum + h[i];
    }

  return sum/100;
  }


void discretize(){

  int blank[9] = {0,0,0,
                  0,0,0,
                  0,0,0};

  if (X > 18.5){
    Xdiscrete = 2;
    }
  else{
    if (X>15){
      Xdiscrete = 1;
      }
    else{
      Xdiscrete = 0;
      }
    }


  if (Y > 17){
    Ydiscrete = 2;
    }
  else{
    if (Y>13.5){
      Ydiscrete = 1;
      }
    else{
      Ydiscrete = 0;
      }
    }

  int index = 3*Ydiscrete + 2 - Xdiscrete;

  blank[index] = 1;
  currentStatus[index] = 1;
  displayPattern(currentStatus);
  checkEntry();
  displayPattern(currentStatus);

  }


void checkEntry(){

  int passwdSum = 0;
  int currentSum = 0;

  for (int i = 0; i < 9; i++){
    passwdSum = passwdSum + password[i];
    currentSum = currentSum + currentStatus[i];}

  for (int i = 0; i < 9; i++){

    if (password[i] == 0){
      if (currentStatus[i] == 1){
        for (int i = 0; i < 9; i++){
         currentStatus[i] = 0;
        }
        delay(100);
        wrongPattern();
        displayPattern(currentStatus);
        return;
        }
      }
    }

  if (passwdSum == currentSum){

    for (int i = 0; i < 9; i++){
      currentStatus[i] = 0;
    }
    delay(100);
    correctPattern();
    return;
    }
  }
```

```
void loop() {

    detect();
    discretize();
    delay(10);

}

void detect(){

  digitalWrite(xTrig,LOW);

  for  (int i=0; i<100;i++){
    digitalWrite(xTrig, LOW);
    delayMicroseconds(2);
    digitalWrite(xTrig, HIGH);
    delayMicroseconds(10);
    digitalWrite(xTrig, LOW);
    duration = pulseIn(xEcho, HIGH);
    xCoords[i] = duration * 0.034 / 2;
  }


  for  (int i=0; i<100;i++){
    digitalWrite(yTrig, LOW);
    delayMicroseconds(2);

    digitalWrite(yTrig, HIGH);
    delayMicroseconds(10);
    digitalWrite(yTrig, LOW);
    duration = pulseIn(yEcho, HIGH);
    // Calculating the distance
    yCoords[i] = duration * 0.034 / 2;
  }

  X = avg(xCoords);
  Y = avg(yCoords);

  Serial.print("X : ");
  Serial.print(X);
  Serial.print(" Y : ");
  Serial.print(Y);
  Serial.println(" ");

  }
```