International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-4, Issue-9)

Research Article | **September 2015**

Knock to Unlock

[1]Biswarup Nandi[*], [2]Soumi Mondal, [3]Ritwik Mutt, [4]Debdeep Das, [5]Rana Biswas
[1] Department of Computer Science and Engineering, University College of Science, Technology and Agriculture, Kolkata, WB, India
[2] Department of Information Technology, Govt. College of Engineering and Leather Technology, Kolkata, WB, India
[3, 4] Department of Computer Science and Engineering, Calcutta Institute of Technology, Howrah, WB, India
[5] Department of Computer Science, St. Xavier's College, Kolkata, WB, India

***Abstract*—**

This The Knock to Unlock is a unique way to unlock a door without a key. It allows the user to program a 'secret' knock sequence that will be the only knock sequence able to unlock the door. It is an interesting way for the user to get to unlock the door and allows the process to be more interactive. Once the secret knock is detected, the door is unlocked and the user can open the door. The Knock to Unlock has specific buttons and lights that allow the user to understand how to use the Knock to Unlock and how the Knock to Unlock is responding to the user. First, there is a button on Top of the door that allows the user to go into 'recording mode'. In this mode the user can record the 'secret' knock sequence that will be the only sequence able to unlock the door. There are also two lights below the button. The red light lights up whenever the user presses the button, indicating the user is in 'recording mode'. The other light is a green light that flashes every time a knock is detected. The Knock to Unlock is a unique way to unlock a door without a key. It allows the user to program a 'secret' knock sequence that will be the only knock sequence able to unlock the door. It is an interesting way for the user to get to unlock the door and allows the process to be more interactive. Once the secret knock is detected, the door is unlocked and the user can open the door. The Knock to Unlock has specific buttons and lights that allow the user to understand how to use the Knock to Unlock and how the Knock to Unlock is responding to the user. First, there is a button on Top of the door that allows the user to go into 'recording mode'. In this mode the user can record the 'secret' knock sequence that will be the only sequence able to unlock the door. There are also two lights below the button. The red light lights up whenever the user presses the button, indicating the user is in 'recording mode'. The other light is a green light that flashes every time a knock is detected. If any mismatch takes place in knocking sequence the owner's mobile will receive a notification.

***Keywords*— *Knocking sequence, Microcontroller, Arduino, GSM300, Lock*.**

I. INTRODUCTION

There are three inputs for the Knock to Unlock. One of the inputs is a piezo element that detects knocks. The user can knock on the front of the door and the piezo element[1] detects these knocks on the other side of the door and sends them to the Arduino. The other inputs are the buttons. The user can push the record button and enter a 'recording mode'. Once this button is released, they are no longer in 'recording mode'. The other button detects if the door is open or closed. If the door is closed the button is being pressed and if the door is open the button is not being pressed Output [servo motor[2] + status LEDs] There are multiple outputs for the Knock to Unlock. The main output is the servomotor. The servomotor or magnetic lock is rotated 90 degrees if the 'secret' knock sequence is detected. This is the prototype's version of 'unlocking' the door. The other outputs are the two LEDs. The red LED is lit up if the button has been pressed, and the green LED is lit up if a knock has been detected.

II. IMPELMENTATION

This project is based on microcontroller. Here we have used Arduino UNO. GSM 300 kit has been used for authentication purpose. Here is the details of implementation.

The Knock to Unlock is implemented on a miniature door replica, with a doorframe and a door that can be opened and closed. There is also a bar that comes down in front of the door, not allowing it to be opened, which simulates our lock. We mounted the piezo element onto the back of the door to detect sounds and the servomotor on the side of the doorframe to rotate the bar in front of the door. We also mounted a button on the inside of the doorframe to detect if the door is opened or not, and finally the record button and LEDs on the top of the doorframe to be visible and accessible by the user. The breadboard and Arduino are mounted just inside the doorframe.

A. Software

The code for the system is all written in the Arduino programming environment. Knock to Unlock The software is used for many of the background processes such as:

- detecting knocks
- recording knocks
- storing knocks in sequences
- comparing knock sequences

© 2015, IJERMT All Rights Reserved

- detecting the state of the system
- unlocking the door
- locking the door
- determining if the door is open or closed
- determining if the door should be locked or unlocked
- lighting up LED'sTitle and Author Details

B. Hardware

Arduino -> Arduino is a single-board microcontroller, intended to make building interactive objects or environments more accessible. It is a tool for making computers that can sense and control more of the physical world than our desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Relay Driver -> Relays have been around for a long time and though often now replaced with solid state switches, they have unique properties that make them more robust than solid-state devices and are not going away. The unique properties are high current capacity, ability to withstand ESD and drive circuit isolation. There are numerous ways to drive relays. In preparation for some of the more advanced relay drivers I will be posting in the future, I have listed a few basic relay drivers for our reference. Included are the following: High side toggle switch driver, low side toggle switch driver, bipolar NPN transistor driver, Darlington transistor driver, N-Channel MOSFET driver, and ULN2003 driver.

Power Supply-> It will take 12v Battery as a power supply to activate the Arduino uno.

Motorized Lock -> Motorized lock installed inside metal doors providing multi-bolt locking. It is Suitable for installation in new or existing metal doors with a multi-point locking mechanism. It is recommended for installation in Internal and external doors in public buildings, institutes and offices, such as - main entrance doors, safety doors with access control, fire doors, emergency and automatic doors.

C. Software Requirements

The Arduino integrated development environment (IDR) is a cross-platform application written in Java, and derives from the IDR for the language and the Wiring projects. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. A program or code written for Arduino is called a sketch. Arduino programs are written is C or C++. The Arduino IDE comes with a software library called "Wiring" from the original Wiring project, which makes many common input/output operations much easier. Users only need define two functions to make a runnable cyclic executive program:

- Setup(): a function run once at the start of a program that can initialize setting
- Loop(): a function called repeatedly until the board powers off.

III. HARDWARE EQUIPMENT

A. Arduino UNO

The Arduino Uno[3] is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

B. BC547B

The NPN Bipolar Transistor[4] is designed for use in linear and switching applications. The device is housed in the TO-92 package, which is designed for medium power applications.

C. Resistor

A resistor is a passive two-terminal electrical component that implements electrical resistance as a circuit element. Resistors act to reduce current flow, and, at the same time, act to lower voltage levels within circuits. It measures by OHM.

D. IN4007

It is a rectifier diode with these features – low forward voltage, high current capability, low leakage current, high surge capability.

E. LED

A light-emitting diode (LED) is a two-lead semiconductor light source. It is a pn-junction diode, which emits light when activated. When a suitable voltage is applied to the leads, electrons are able to recombine with electron holes within the device, releasing energy in the form of photons. This effect is called electroluminescence, and the color of the light (corresponding to the energy of the photon) is determined by the energy band gap of the semiconductor.

© 2015, IJERMT All Rights Reserved

F. Relay

Relay[5] is an electromagnetic device which is used to isolate two circuits electrically and connect them magnetically. They are very useful devices and allow one circuit to switch another one while they are completely separate. They are often used to interface an electronic circuit (working at a low voltage) to an electrical circuit which works at very high voltage. For example, a relay can make a 5V DC battery circuit to switch a 230V AC mains circuit.

G. Piezoelectric Sensor

A piezoelectric sensor is a device that uses the piezoelectric effect, to measure changes in pressure, acceleration, strain or force by converting them to an electrical charge

H. Electro-magnetic Lock

An electromagnetic lock[6], magnetic lock, or maglock is a locking device that consists of an electromagnet and an armature plate. There are two main types of electric locking devices. Locking devices can be either "fail safe" or "fail secure". A fail-secure locking device remains locked when power is lost. Fail-safe locking devices are unlocked when de-energized. Direct pull electromagnetic locks are inherently fail-safe. Typically the electromagnet portion of the lock is attached to the door frame and a mating armature plate is attached to the door. The two components are in contact when the door is closed. When the electromagnet is energized, a current passing through the electromagnet creates a magnetic flux that causes the armature plate to attract to the electromagnet, creating a locking action. Because the mating area of the electromagnet and armature is relatively large, the force created by the magnetic flux is strong enough to keep the door locked even under stress.

I. GSM SIM 300

A GSM Modem is a device that modulates and demodulates the GSM signals and in this particular case 2G signals. The modem we are using is SIMCOM SIM300. It is a Tri-band GSM/GPRS Modem as it can detect and operate at three frequencies (EGSM 900 MHz, DCS 1800 MHz and PCS1900 Mhz). Default operating frequencies are EGSM 900MHz and DCS 1800MHz.

Sim300[7] GSM module used here, consists of a TTL interface and an RS232 interface. The TTL interface allows us to directly interface with a microcontroller while the RS232 interface includes a MAX232 IC to enable communication with the PC. It also consists of a buzzer, antenna and SIM slot. Sim300 in this application is used as a DCE (Data Circuit-terminating Equipment) and PC as a DTE (Data Terminal Equipment).

IV. ALGORITHM & DFD

A. Algorithm

```
/* Detects patterns of knocks and triggers a motor to unlock
it if the pattern is correct.
Analog Pin 0: Piezo speaker (connected to ground with 1M pulldown resistor)
Digital Pin 2: Switch to enter a new code.  Short this to enter programming mode.
Digital Pin 3: DC gear reduction motor attached to the lock. (Or a motor controller or a solenoid or other unlocking
mechanisim.)
Digital Pin 4: Red LED.
Digital Pin 5: Green LED.
Digital Pin 6: Yellow LED */
    Step – 0 :
    //Pin definitions
    knockSensor = A0;        // Piezo sensor on pin 0.
    programSwitch = 2;       // If this is high we program a new code.
    lockMotor = 3;           // Gear motor used to turn the lock.
    redLED = 4;              // Status LED
    greenLED = 5;            // Status LED
    yellowLED = 5;           // Status LED (GSM)
    Step – 1 :
    // Tuning constants.  Could be made vars and hoooked to potentiometers for //soft configuration, etc.
    threshold = 3;           // Minimum signal from the piezo to register as
                                          a knock
    rejectValue = 25;        // If an individual knock is off by this
                                          percentage of a knock we don't unlock..
    averageRejectValue = 15; // If the average timing of the knocks is off
                                          by this percent we don't unlock.
    knockFadeTime = 150;     // milliseconds we allow a knock to fade before
                                          we listen for another one. (Debounce timer.)
    lockTurnTime = 650;      // milliseconds that we run the motor to get it
                                          to go a half turn.
    maximumKnocks = 20;      // Maximum number of knocks to listen for.
```

© 2015, IJERMT All Rights Reserved

```
knockComplete = 1200;     // Longest time to wait for a knock before we
                                        assume that it's finished.
Step – 2 :
// Variables.
secretCode[maximumKnocks] = {50, 25, 25, 50, 100, 50, 0, 0, 0, 0, 0, 0,
                                        0, 0, 0, 0, 0, 0, 0, 0};
                                        // Initial setup: "Shave and a Hair Cut, two bits."
knockReadings[maximumKnocks];   // When someone knocks this array fills
                                        with delays between knocks.
knockSensorValue = 0;          // Last reading of the knock sensor.
programButtonPressed = false;   // Flag so we remember the programming
                                                button setting at the end of the cycle.
boolean GSMError = TRUE;
Step – 3 :
//setup()
pinMode(lockMotor, OUTPUT);
pinMode(redLED, OUTPUT);
pinMode(greenLED, OUTPUT);
pinMode(yellowLED, OUTPUT);
pinMode(programSwitch, INPUT);
Serial.begin(9600);                    // Uncomment the Serial.bla lines for
                                                debugging.
Serial.println("Program start.");  // but feel free to comment them out
                                                after it's working right.
testGSMKit();
digitalWrite(greenLED, HIGH);      // Green LED on, everything is go.
Step – 4 :
//loop()
// Listen for any knock at all.
knockSensorValue = analogRead(knockSensor);
        if (digitalRead(programSwitch)==HIGH){  // is the program button
                                                        pressed?
                programButtonPressed = true;        // Yes, so lets save
                                                        that state
                digitalWrite(redLED, HIGH);         // and turn on the red
                                                        light too so we know we're
                                                programming.
        } else {
                programButtonPressed = false;
                digitalWrite(redLED, LOW);
        }
        if (knockSensorValue >=threshold){
                listenToSecretKnock();
        }
Step – 5 :
// Records the timing of knocks.
//listenToSecretKnock()
Serial.println("knock starting");
// First lets reset the listening array.
for (i=0;i<maximumKnocks;i++){
        knockReadings[i]=0;
}
currentKnockNumber=0;                   // Incrementer for the
                                                array.
startTime=millis();                     // Reference for when this
                                                knock started.
now=millis();
digitalWrite(greenLED, LOW);            // we blink the LED for
                                                a bit as a visual indicator of the
                                                knock.
if (programButtonPressed==true){
        digitalWrite(redLED, LOW);      // and the red
                                                one too if we're programming a
                                        new knock.
```

© 2015, IJERMT All Rights Reserved

```
}
delay(knockFadeTime);                    // wait for
                                                    this peak to fade before we
                                                listen to the next one.
digitalWrite(greenLED, HIGH);
if (programButtonPressed==true){
        digitalWrite(redLED, 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.
Serial.println("knock.");
now=millis();
knockReadings[currentKnockNumber] = now-startTime;
currentKnockNumber ++;              //increment the counter
startTime=now;
// and reset our timer for the next knock
         digitalWrite(greenLED, LOW);
if (programButtonPressed==true){
        digitalWrite(redLED, LOW);   // and the red one too if we're
                                                programming a new knock.
}
delay(knockFadeTime);    // again, a little delay to let the knock
                                                decay.
digitalWrite(greenLED, HIGH);
if (programButtonPressed==true){
        digitalWrite(redLED, 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
if (programButtonPressed==false){          // only if we're not in
                                                    progrmaing mode.
        if (validateKnock() == true){
        triggerDoorUnlock();
        } else {
                GSMSendMsg("<your message>", "0000000000"); //OWNER's Mobile
                                                number where notification for error entry will be delivered
        Serial.println("Secret knock failed.");
        digitalWrite(greenLED, LOW);              // We didn't unlock, so
                                                    blink the red LED as visual
                                                    feedback.
        for (i=0;i<4;i++){
                digitalWrite(redLED, HIGH);
                delay(100);
                digitalWrite(redLED, LOW);
                delay(100);
        }
        digitalWrite(greenLED, HIGH);
        }
} else { // if we're in programming mode we still validate the lock, we
                just don't do anything with the lock
        validateKnock();
                                // and we blink the green and red alternately to show that program is
                complete.
        Serial.println("New lock stored.");
        digitalWrite(redLED, LOW);
```

© 2015, IJERMT All Rights Reserved

```
                digitalWrite(greenLED, HIGH);
                for (i=0;i<3;i++){
                        delay(100);
                digitalWrite(redLED, HIGH);
                digitalWrite(greenLED, LOW);
                delay(100);
                digitalWrite(redLED, LOW);
                digitalWrite(greenLED, HIGH);
                        }
                }
        }
        Step – 6:
        // Runs the motor (or whatever) to unlock the door.
        //triggerDoorUnlock()
        Serial.println("Door unlocked!");
        // turn the motor on for a bit.
        digitalWrite(lockMotor, HIGH);
        digitalWrite(greenLED, HIGH);          // And the green LED too.
        delay (lockTurnTime);                  // Wait a bit.
        digitalWrite(lockMotor, LOW);          // Turn the motor off.
        // Blink the green LED a few times for more visual feedback.
        for (i=0; i < 5; i++){
        digitalWrite(greenLED, LOW);
        delay(100);
        digitalWrite(greenLED, HIGH);
        delay(100);
                }
        }
        Step – 7 :
        // 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.
        //validateKnock()
        // simplest check first: Did we get the right number of knocks?
        currentKnockCount = 0;
        secretKnockCount = 0;
        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 (programButtonPressed==true){
                for (i=0;i<maximumKnocks;i++){ // normalize the times
        secretCode[i]= map(knockReadings[i],0, maxKnockInterval, 0, 100);
        }
    // And flash the lights in the recorded pattern to let us know it's
        been programmed.
        digitalWrite(greenLED, LOW);
        digitalWrite(redLED, LOW);
        delay(1000);
        digitalWrite(greenLED, HIGH);
        digitalWrite(redLED, HIGH);
        delay(50);
```

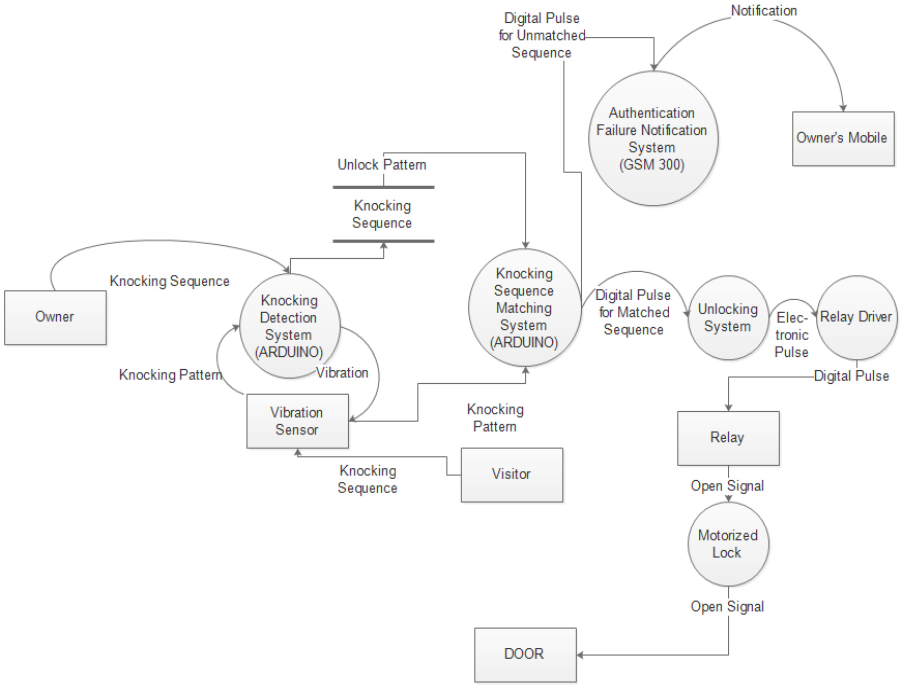
© 2015, IJERMT All Rights Reserved

```
        for (i = 0; i < maximumKnocks ; i++){
                digitalWrite(greenLED, LOW);
        digitalWrite(redLED, LOW);
        // only turn it on if there's a delay
        if (secretCode[i] > 0){
                delay( map(secretCode[i],0, 100, 0, maxKnockInterval));
                                                        //Expand the time back out to what it was.
                                                    Roughly.
        digitalWrite(greenLED, HIGH);
        digitalWrite(redLED, HIGH);
        }
        delay(50);
}
         return false;        // We don't unlock the door when we are recording a new
                                    knock.
        }
        if (currentKnockCount != secretKnockCount){
                return false;
        }
        Step – 8:
        /*  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.
          */
        totaltimeDifferences=0;
        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]);
        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;
        }
        Step – 9 :
        //GSMSendMsg(String msg, String num)
        if(testGSMKit()) {
                digitalWrite(yellowLED, LOW);
                Serial.println("AT+CMGS=\""+num+"\"");
                digitalWrite(yellowLED, HIGH);
                delay(100);
                digitalWrite(yellowLED, LOW);
                while(Serial.read()!='>');
                digitalWrite(yellowLED, HIGH);
                delay(500);
                digitalWrite(yellowLED, LOW);
                Serial.print(msg);
                Serial.write(0x1A);
                digitalWrite(yellowLED, HIGH);
                Serial.write(0x0D);
                Serial.write(0x0A);
                digitalWrite(yellowLED, LOW);
                delay(2000);
                digitalWrite(yellowLED, HIGH);
                }
```

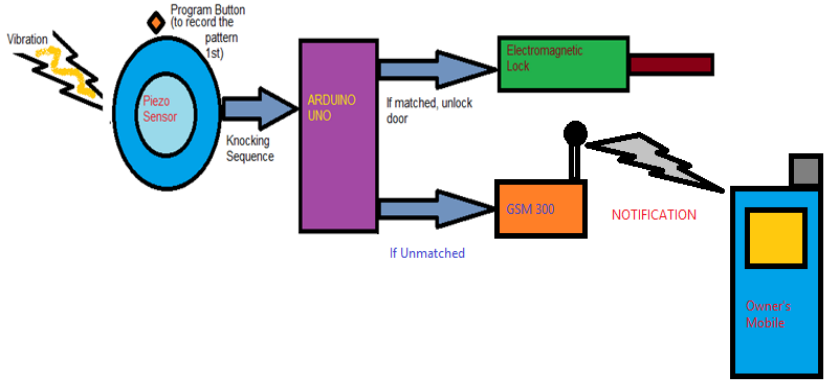
© 2015, IJERMT All Rights Reserved

```
}
Step – 10 :
//boolean testGSMKit(){
digitalWrite(yellowLED, LOW);
delay(2000);
Serial.println("AT");
delay(2000);
if(Serial.readString().indexOf("OK")== -1) {
  GSMError=true;
  return false;
}
digitalWrite(yellowLED, HIGH);
if (GSMError) {
GSMError=false;
delay(2000);
digitalWrite(yellowLED, LOW);
Serial.println("AT+CMGF=1");
delay(2000);
digitalWrite(yellowLED, HIGH);
Serial.println("AT+CLIP=1");
}
return true;
}
Step -  11 : END
```

B. Data Flow Diagram

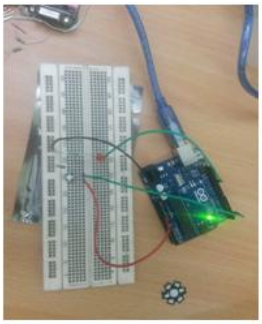


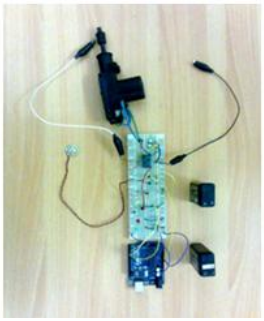
V. BLOCK DIAGRAM



© 2015, IJERMT All Rights Reserved

VI. SNAPSHOTS





VII. CONCLUSIONS

Our project has several advantages, like, no need to keep key for a particular lock; cost efficient than other high-end lock; low operating cost; locking technique is unique; low cost, satisfiable quality and less maintenance and robust device. Even we have installed a GSM based authentication such that if any mismatch occurs in knocking sequence notification will be sent to owner's mobile.

In future we would try to upgrade this lock in terms of GSM controlling such that lock and unlock can be done using GSM devices.

ACKNOWLEDGMENT

We would like to thank our project guide Mr. Rana Biswas for helping us in this form. We also thankful to the institutions : University College of Science, Technology and Agriculture, Govt. College of Engineering and Leather Technology and Calcutta Institute of Technology for providing us with the opportunity to develop and exhibit the project.

REFERENCES

[1] PIEZOELECTRIC TECHNOLOGY PRIMER by James R. Phillips Sr. Member of Technical Staff CTS Wireless Components 4800 Alameda Blvd. N.E. Albuquerque, New Mexico 87113.
[2] SERVO CONTROL SYSTEMS 1 : DC Servomechanism by Elke Laubwald: Visiting Consultant, control systems principle.co.uk
[3] https://www.arduino.cc/en/Main/arduinoBoardUno
[4] S Salivahanan and V.S. Kanchana Bhaaskaran "Linear Integrated Circuits",ISBN 13:978-0-07-064818-0, p. 19.
[5] "Boolean Logic to Switching Circuits and Automata: Towards Modern Information Technology" By Radomir S. Stankovic, Jaakko Astola , e-ISBN 13:978-3-64, p. 160.
[6] "Illustrated Guide to Door Hardware: Design, Specification, Selection" By Scott Tobias, Wiley Publication, p. 170.
[7] http://www.engineersgarage.com/contribution/how-to-interface-GSM-SIM-300-modem-with-atmega32-to-send-and-receive-SMS

© 2015, IJERMT All Rights Reserved