LibSafe+
The Smart Alarm for Your Device
ECE 110/120 Honors Lab

Joshua Sanchez – jsanch84
Yulun Yao – yuluny2
Wentao Jiang – wentaoj2
Introduction

When you walk into Grainger Library, you'll see a sign that warns of high theft, particularly of laptops. You might also see similar signs in libraries at other universities or municipalities. The ubiquity of these signs in libraries is indicative of how common the issue of laptop theft is in these establishments. Current software-based solutions, such as VirtuaLock and LAlarm, require network connections and active computer interactions. Current hardware-based solutions, such as the Targus DEFCON 1 system, are expensive. LibSafe+ aims to be an affordable and effective theft alert and deterrent system that forgoes the issues of software-based alarms by using efficiently designed hardware to detect theft and alert the user and those around the area of theft.

LibSafe+ uses sensors and a password system to activate an alarm when laptops leave designated, locked surfaces. The final password system features a four-digit password and 7 different password input buttons, for a total of 2401 different password combinations. Compared to other security solutions, LibSafe+ is cheaper and portable.

Design

The device is comprised of two stand-alone systems, an alarm system and a password system. To activate the device, a user puts their laptop on it. This action activates an ultrasonic sensor and accelerometer sensor within the device. These sensors are used to detect whether the laptop has been removed from the device, or the device has been removed with the stolen laptop. Should these things occur, the Arduino unit activates an alarm buzzer and LED that signify a theft. The alarm system receives an input signal from the password system to determine whether the device is locked. If the signal signifies that the system is unlocked, no alarms are triggered.
The password system consists of several logic components. The Input-Password Equivalence Controller and input logic is used to select password digit signals with which to compare button inputs. Since the design of the password system is based on the finite state machine model, the results of the comparison are used to determine the next state of the system. A reset controller is used to detect when password digits are being reset and makes sure that inputs go to correct storage units.

In the alarm system implementation, the ultrasonic sensor requires at least 2.5cm from the nearest object to function correctly. Thus, the sensor goes at the bottom of the device so that it can measure an appropriate distance. In terms of the sensitive accelerometer sensor, our vertical acceleration threshold is 1.15 so that the alarm does not accidentally activate when the user is, for example, typing in a password or activating the device. The buzzer and LEDs require separate output pins due to the frequency of which each individual component is used in our code.

*Figure 2. Prototype Block Diagram of Password System.*
Results

An ultrasonic sensor functions as a distance sensor. There is an ultrasonic transmitter and a receiver on this sensor. The sensor first emits ultrasonic waves. When a wave encounters the obstacle, it will come back and be received by a receiver on the sensor. We can measure the time the wave travels to measure the distance between the obstacle and the sensor, which we would call the “pulse width.” The way we do it is to first send a high voltage to the trig pin on the sensor for a moment to emit a wave. At the same time, we start to clock the time the echo pin, or receiver, sends back low voltage. When the wave hits the receiver, the echo pin would return high voltage, and the clock stops.

We use Matlab to generate an equation for the relationship:

\[ y = p1 \times x + p2 \]

Coefficients:

\[ y = \text{Pulse Width(μs)} \]
\[ x = \text{Actual Distance(cm)} \]
\[ p1 = 57.006 \]
\[ p2 = 7.7333 \]
The accelerometer sensor we used is MMA8542Q. In order to facilitate the process of construction, we used the official Arduino library for this sensor. We followed the instruction to set the sensor up. Using the command “accel.read”, the Arduino starts to read the value from the sensor. The value we need is “accel.cz”, which is the calculated vertical acceleration.

We first tested the alarm system alone. We put the device into a small box as shown in figure 2. We then put a laptop on the box, and connected the power. The system didn’t alarm because of that the laptop was on it. We then took away the laptop, and the system started to alarm.

We then disconnected the power and put the laptop onto the box again and connected the power. This time, we tried to take both the device and the laptop, and the device started to alarm.

Since the password was yet to finish, we just simulated the two possible input from the password system - low voltage(0) and high voltage(1). When we didn’t input any voltage(low voltage) to the receiving pin, the alarm system would work. When we input a voltage, the alarm system wouldn’t work when we took away the laptop or took the two things together.

The three tests proved that the device would work as we had expected.

**Problems and Challenges**

As a group, we felt overwhelmed in the first couple of weeks. Our group did not yet have much experience with programming and circuitry. Since we did not learn about important concepts related to our designs until around the middle of the semester, it made it hard to get our designs done in time to create a full, working prototype. Later on, it became hard to find designs for our prototype that we would be able to implement within a reasonable timeframe, and so the concern became designing a reasonable prototype that still maintained the features we wanted to showcase. In addition, a key issue in our presentations of the prototype was a circuit problem we kept encountering with our sensors. Sometimes, the sensors would not work until we either changed their position on the breadboard or removed and reinserted multiple times.

**Future Plans**

Since we have finished the alarm system, we will try to make a PCB version of it. We hope that the PCB version will have no issue of the loose connection of the sensors. It is too easy for the components to be disconnected from the breadboards.

We would make a case for the alarm system, with the LEDs on the sides of the box. The buzzer will be on the top side. We may will also record a voice saying “Alert! Theft!” to replace the noise emitted by the buzzer. We will leave a hole on the top side of the box for the ultrasonic sensor to detect the distance to the laptop. The PCB will be fixed firmly onto the bottom of the box to ensure that the sensor is right under the hole. We would also add a button to turn the Arduino on and off on the bottom of the device. The password system will be in another box connected to the alarm system box. It will include the full implementation of 7 buttons for password input and a reset button.
Appendix

1) Arduino Code

#include <Wire.h>
#include <SFE_MMA8452Q.h>

MMA8452Q accel;
const int TRIG_PIN = 7;
const int ECHO_PIN = 8;
const unsigned int MAX_DIST = 23200;
unsigned long t1;
unsigned long t2;
unsigned long pulse_width;
float cm;
int timer = 0;
int state = 0;
int val = 0;

void setup() {
    accel.init();
    pinMode(TRIG_PIN, OUTPUT);
    digitalWrite(TRIG_PIN, LOW);
    pinMode(13, OUTPUT);
    pinMode(11, OUTPUT);
    pinMode(2, INPUT);
}

void alarm() {
    if (cm > 5 || accel.cz > 1.15) {
        for (int t; t < 1000; t++) {
            digitalWrite(11, HIGH);
            delay(5);
            digitalWrite(11, LOW);
            delay(5);
            if (timer < 50) timer++;
        }
    }
}
else {
    if (state == 100) state = 0;
    if (state % 2 == 0) digitalWrite(13, HIGH);
    else digitalWrite(13, LOW);
    timer = 0;
    state++;
}

void loop() {
    val = digitalRead(2);
    if (val == LOW) {
        digitalWrite(TRIG_PIN, HIGH);
        delayMicroseconds(10);
        digitalWrite(TRIG_PIN, LOW);
        while (digitalRead(ECHO_PIN) == 0);
        t1 = micros();
        while (digitalRead(ECHO_PIN) == 1);
        t2 = micros();
        pulse_width = t2 - t1;
        cm = (pulse_width - 7.7333) / 57.006;
        if (accel.available()) {
            accel.read();
        }
        alarm();
    }
    if (val == HIGH) {}
References

