Suite\_Lock

By:

Chris Bero

12-11-2012

EE 100

The SuiteLock project is intended to make entry and egress from a Frank Franz Residence Hall dormitory more efficient and secure.

Method of Operation:

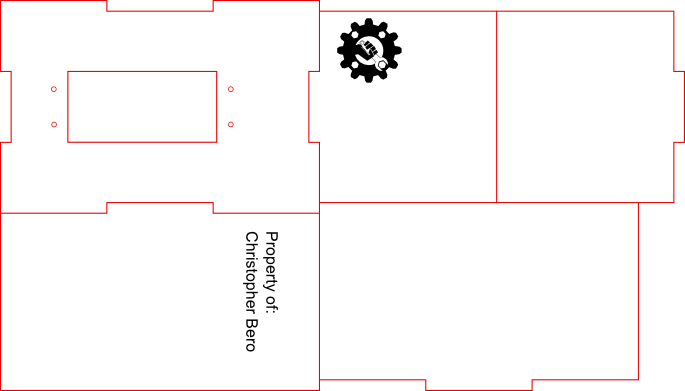
A Charger Card is swiped on a card reader affixed to the door’s exterior. The card reader acts as an ASCII keyboard, directly printing the card’s data to the computer as well as appending a line return. The computer receives the data and directs it into the SuiteLock server program, written in C++. The program takes the input and compares it to known-validated-users. If a valid user is detected, the program will use the COM6 serial connection to send [U] to a Diavolino. The Diavolino will then take the [U] out of buffer and determine that it is [U]. Once validated on the Diavolino’s side, the microcontroller will call a function which will control a servo to unlock the door. The servo is mounted inside of a custom-made, laser-cut enclosure and glued above the deadbolt. Attached to the servo is a custom made “arm” which toggles the deadbolt, while still leaving room for the deadbolt to be turned by a key.

The microcontroller is equipped with a WatchDogTimer, which will hopefully prevent the controller from freezing or halting indefinitely.

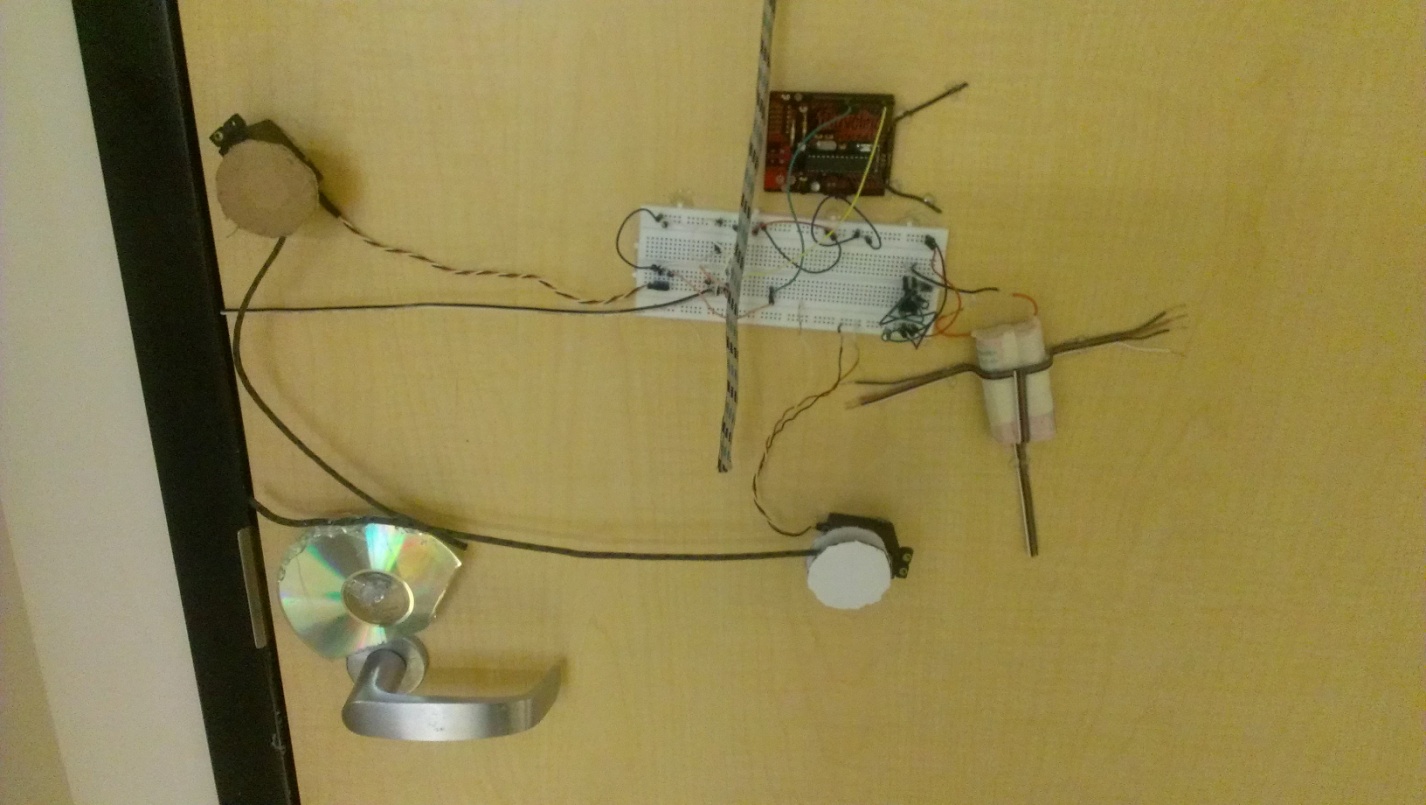
Theory of Operation:

A user (one of four roommates) is required to use a Charger Card to access the building’s lobby. Normally a key is used to access the common area of the dorm and individual rooms. The SuiteLock uses the roommate’s Charger Card to access the common area. Under normal circumstances, the elimination of a unique security step (a key and a card, instead of a card and a card) would not help with security practices. This adjustment must be taken in context. Without SuiteLock, the common door is generally left unlocked, because most roommates do not care to spend the time to both unlock the door to the dorm, and then the door to their room. Thus, having a system that both automatically locks the dorm and is easier to unlock than a deadbolt has an advantage. In addition, the two step process of cards and keys is in effect in order to reach a roommates bedroom.

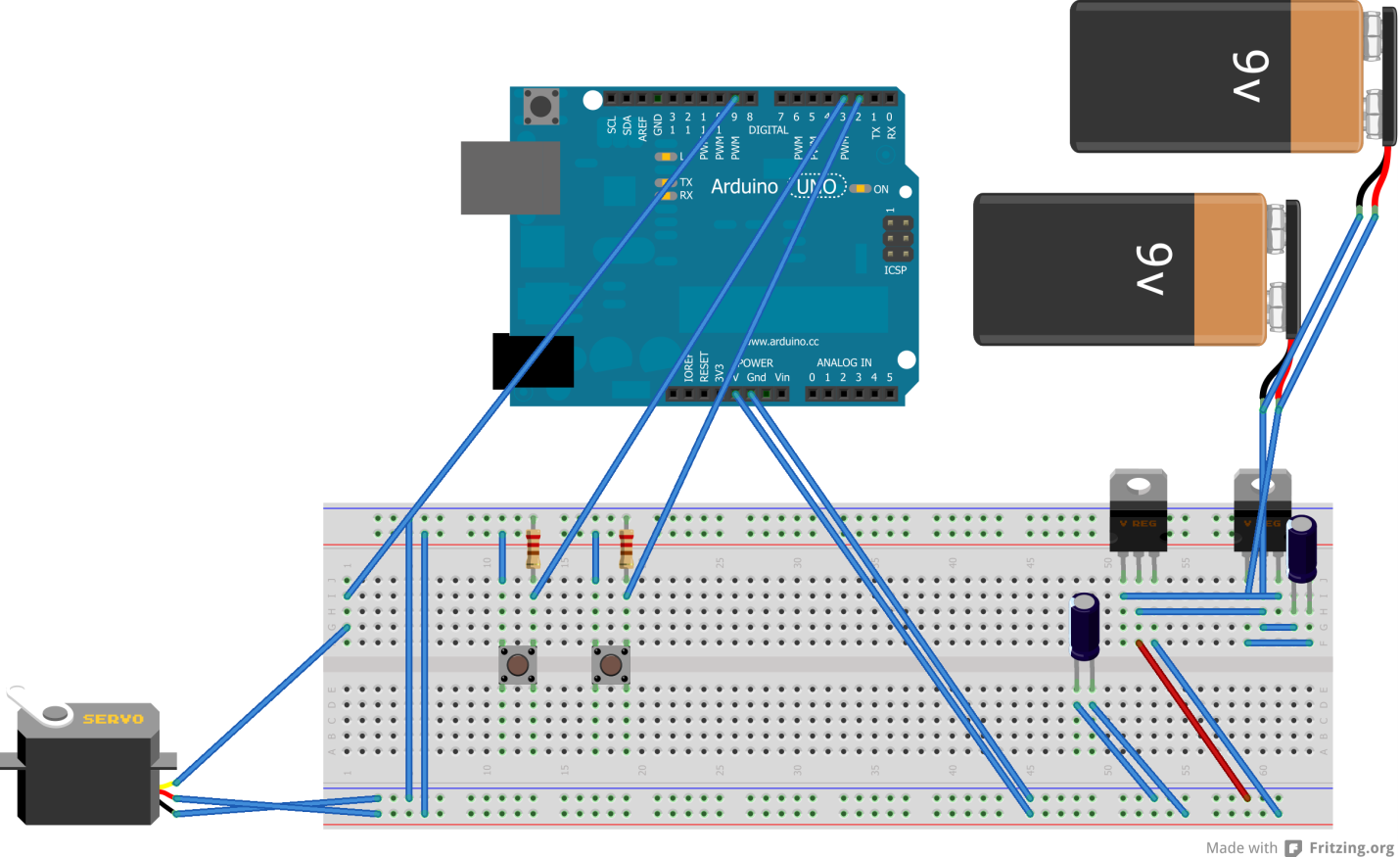
The servo mount:



The first revision of the door lock system:



A simplified rendering of the circuit before design:



Arduino Sketch

//

// SuiteLock v.2.1b

// By: Chris Bero (bigbero@gmail.com)

// Last Updated: 11.26.2012

//

#include <Servo.h>

#include <avr/wdt.h>

// Pin Constants:

const int servoPin = 9;

const int doorbtn = 3;

const int bitchbtn = 2;

const int lockbtn = 4;

const int doorLed = 5;

const int servDelay = 600; // The delay allowing for the servo to complete an action

//Variables:

int doorState = 0; // The value returned by the door button (0 or 1)

int servState = 90; // The position of the servo in degrees (0 thru 180)

int bitchState = 0; // Value returned by the bitch button (0 or 1)

int lockbtnState = 0;

int index = 0;

int serialByte = 0; // Reads in a single character from the serial connection

byte lastSerial = 0;

int smallBlink = 0;

int statusDelay = 2000;

float cps = 0.0;

unsigned long prevMillis = 0;

unsigned long progCycles = 0;

unsigned long cycleCount = 0;

unsigned long bitchTime = 0; // Delay on locking the dooor

char inData[20] = {0}; // Compiled serial data into a string for comparison

char lastData[20] = {0};

char\* Chris = "000000000";

char\* ChrisPin = "2139";

char\* Luc = "000000000";

char\* LucPin = "2222";

char\* Andrew = "000000000";

char\* AndrewPin = "4321";

bool dostatus = true; // Determine whether to send sys status

Servo serv;

// Set up the environment

void setup()

{

wdt\_enable(WDTO\_2S);

pinMode(doorbtn, INPUT);

pinMode(bitchbtn, INPUT);

pinMode(lockbtn, INPUT);

pinMode(doorLed, OUTPUT);

pinMode(servoPin, OUTPUT);

serv.attach(servoPin);

Serial.begin(9600);

prevMillis = millis();

}

////////////////////////////////////////////////

// Statuser - Sends system status to Serial

/////////////////////////////////////////////

int statuser ()

{

Serial.println("[Start]"); //Start Of Transmission

refresh();

unsigned long currentSeconds = (millis()/1000); // Cumulative time

Serial.print("-Time Alive: ");

int sc = currentSeconds % 60;

currentSeconds /= 60;

int mn = currentSeconds % 60;

currentSeconds /= 60;

int hr = currentSeconds % 24;

int dy = currentSeconds / 24;

Serial.print(dy);

Serial.print(":");

Serial.print(hr);

Serial.print(":");

Serial.print(mn);

Serial.print(":");

Serial.println(sc);

Serial.print("-Avg Cycles per Second: ");

Serial.println(cps, 4);

Serial.print("-DoorState: ");

Serial.println(doorState);

Serial.print("-Bitch Seconds: ");

if (bitchTime == 0)

{

Serial.println("Not Active"); // The timer isn't running

}

else

{

Serial.println((bitchTime - millis()) / 1000); // Print the number of seconds left before the door locks

}

Serial.print("-Servo Position: ");

Serial.println(servState);

Serial.print("-Last Serial Byte: [");

if (lastSerial == '\r')

{

Serial.print("Return");

}

else

{

Serial.write(lastSerial);

}

Serial.println("]");

Serial.print("-Last Serial Array[");

for (int c = 0; c < 20; c++)

{

if (lastData[c] == '\0')

{

break;

}

Serial.write(lastData[c]);

}

Serial.print("]");

Serial.println("\n\r[End]\n\r"); //End Of Transmission

return(0);

}

////////////////////////

// Lock the door.

/////////////////////

int locker()

{

// Check the button states.

refresh();

// Make sure the door is closed.

do

{

refresh();

delay(500);

}

while(doorState == LOW);

// Turn on the locking LED during the servo movement.

digitalWrite(doorLed, HIGH);

wdt\_reset();

// Tell the servo to turn to 20 degrees.

serv.write(20);

// Give the servo time to complete the turn.

delay(servDelay);

wdt\_reset();

// Turn the servo opp direction to reset.

serv.write(90);

// Wait for the servo to reach it's reset point.

delay(servDelay);

// Turn off the cool little LED.

digitalWrite(doorLed, LOW);

// Call parents for 11pm checkup

return(0);

}

/////////////////////////

// Unlock the door.

//////////////////////

int unlocker ()

{

// Check the pin states.

refresh();

// Turn on the status LED.

digitalWrite(doorLed, HIGH);

wdt\_reset();

// Turn servo to 170 degrees to unlock the door.

serv.write(170);

// Wait for servo motion to complete.

delay(servDelay);

wdt\_reset();

// Reset the servo to 90 degrees.

serv.write(90);

// Wait for reset motion to complete.

delay(servDelay);

// Turn off LED.

digitalWrite(doorLed, LOW);

return(0);

}

////////////////////////////////////////////

// Compiles and verifies user passkeys

//////////////////////////////////////////

void incomingSerial ()

{

refresh();

index = 0;

int whileLooper = 0;

while (index < 20 && whileLooper < 50)

{

refresh();

int serialHold = 0;

while ((Serial.available() == 0) && (serialHold < 30))

{

refresh();

delay(100);

serialHold++;

}

if (Serial.available() > 0)

{

// Reset the serialByte, done for debugging.

serialByte = 0;

wdt\_reset();

// Read the serial byte.

serialByte = Serial.read();

lastSerial = serialByte;

}

if ((serialByte != ';') && (serialByte != '=') && (serialByte != '.') && (serialByte != 0))

{

inData[index] = serialByte;

index++;

serialByte = 0;

}

else if ((serialByte == '=') || (serialByte == '.'))

{

inData[index] = '\0';

break;

}

whileLooper++;

}

if (serialByte == '=')

{

if (strcmp(inData, Chris) == 0)

{

unlocker();

Serial.println("Unlocking for Chris.");

}

else if (strcmp(inData, Andrew) == 0)

{

unlocker();

Serial.println("Unlocking for Andrew.");

}

else if (strcmp(inData, Luc) == 0)

{

unlocker();

Serial.println("Unlocking for Luc.");

}

}

else if (serialByte == '.')

{

if (strcmp(inData, ChrisPin) == 0)

{

unlocker();

Serial.println("Unlocking for Chris.");

}

else if (strcmp(inData, AndrewPin) == 0)

{

unlocker();

Serial.println("Unlocking for Andrew.");

}

else if (strcmp(inData, LucPin) == 0)

{

unlocker();

Serial.println("Unlocking for Luc.");

}

}

for (int c = 0; c < 20; c++)

{

lastData[c] = '\0';

}

for (int c = 0; c < index; c++)

{

Serial.print(inData[c]);

lastData[c] = inData[c];

}

for (int c = 0; c < 20; c++)

{

inData[c] = '\0';

}

index = 0;

}

///////////////////////////////

// Refresh button states.

/////////////////////////////

void refresh ()

{

wdt\_reset();

doorState = digitalRead(doorbtn);

bitchState = digitalRead(bitchbtn);

lockbtnState = digitalRead(lockbtn);

servState = serv.read();

}

///////////////////////

// Main function.

////////////////////

void loop()

{

wdt\_reset();

// Blink the LED every so many cycles.

if (smallBlink == 5)

{

smallBlink = 0;

if (doorLed == LOW)

{

digitalWrite(doorLed, HIGH);

}

else

{

digitalWrite(doorLed, LOW);

}

}

// Status

if(dostatus == true)

{

unsigned long timeCount = (millis() - prevMillis);

if (timeCount > statusDelay)

{

//Serial.print("Cycles: ");

//Serial.println(cycleCount);

cps = (cycleCount/((float)timeCount/1000)); // Calculate a x second average for Cycles Per Second

statuser(); // Print Status to serial connection

cycleCount = 0; // Reset the cycle Counter

prevMillis = millis(); // Reset the statuser interval

}

}

// Refresh button states.

refresh();

if (bitchTime < millis() && bitchTime != 0)

{

// Tyler's Bitch Button isn't active anymore, fix it.

bitchTime = 0;

locker();

}

if (bitchState == HIGH)

{

while (bitchState == HIGH)

{

refresh();

delay(200);

}

if (bitchTime == 0)

{

bitchTime = millis(); // Basically zero

}

bitchTime += 600000; // Add ten minutes

if ((bitchTime - millis()) > 1800000) // If the delay is greater than 30 mins

{

bitchTime = millis() + 1800000; // Set it equal to 30 mins

}

}

// Is the door closed and not locked? \*Gasp\*

if ((doorState == LOW))

{

if (bitchTime > millis())

{

// Tyler's Bitch Button is active, wait.

}

else

{

// Fix it.

while (doorState == LOW)

{

wdt\_reset();

delay(100);

refresh();

if (bitchState == HIGH) // This checks to see if some idiot opened the door before pressing the button

{

while (bitchState == HIGH)

{

refresh();

delay(50);

}

if (bitchTime == 0)

{

bitchTime = millis(); // Basically zero

}

bitchTime += 600000; // Add ten minutes

if ((bitchTime - millis()) > 1800000) // If the delay is greater than 30 mins

{

bitchTime = millis() + 1800000; // Set it equal to 30 mins

}

}

refresh();

if (lockbtnState == 1)

{

locker();

}

}

//We're yielding to Tyler's unplugging, no more auto-lock :(

//locker();

}

}

refresh();

if (lockbtnState == 1)

{

locker();

}

// Check for available communications.

if (Serial.available() > 0)

{

// Reset the serialByte, done for debugging.

serialByte = 0;

wdt\_reset();

// Read the serial byte.

serialByte = Serial.read();

lastSerial = serialByte;

}

// Act on the byte data.

if (serialByte == ';' || serialByte == '.')

{

incomingSerial();

}

else if (serialByte == 'U')

{

// Let someone in.

unlocker();

// Wait for the door to change states.

delay(servDelay);

}

else if (serialByte == 'L')

{

locker();

delay(servDelay);

}

else if (serialByte == 'S')

{

statuser();

delay(servDelay/10); // don't want to wait around for 600ms

}

Serial.flush(); /\* I have no idea why this is here \*/

// Clean serialByte for debugging.

serialByte = 0;

smallBlink++;

cycleCount++;

}

Unused code - AutoHotKey

; =========================================

; SuiteLock

; By: Christopher Bero

; v.2.1a

; Last Updated: 10.24.2012

; =========================================

; This program will interpret information read from Charger cards via the card reader

; and will dictate whether the door should be unlocked to the Arduino by USB Serial connection.

; Set Global Configs

SetTitleMatchMode, 2

#NoEnv ; Recommended for performance and compatibility with future AutoHotkey releases.

SendMode Input ; Recommended for new scripts due to its superior speed and reliability.

SetWorkingDir %A\_ScriptDir% ; Ensures a consistent starting directory.

OnExit, SubShutdown

; Set Global Variables

PickSound := 0

serialData := 0

Serials := 0

serialStart = SOT

serialEnd = EOT

serialHolder =

;=====================================

; Start of program, autostarts without hotkey

;=====================================

::

setup()

;======================

; Initiates one-time items

;======================

setup()

{

;Gui, Add, Text, W300 R10 vSerials, Serial Monitor

;Gui, Show

;Gui +resize +AlwaysOnTop

main()

}

;======================

; Plays a particular audio file, still in beta stage

;======================

playSound(type)

{

if (type = "bell")

{

}

else if (type = "error")

{

Random, PickSound, 1, 3

Run, audio\error\%PickSound%.mp3, , Min UseErrorLevel

}

else if (type = "admit" || type = "generic")

{

Random, PickSound, 1, 3

Run, audio\admit\%PickSound%.mp3

}

else if (type = "deny")

{

Random, PickSound, 1, 7

Run, audio\deny\%PickSound%.mp3

}

else

{

Run, audio\users\%type%.mp3

}

}

;=======================

; The main function

;=======================

main()

{

While (1)

{

;Wait for any general input, end capture string with a return character

Input, password, I,{Enter},

;If the above input was indeed terminated with a return, search for matches

ifInString, ErrorLevel, EndKey:

{

; serial settings

ARDUINO\_Port = COM6

ARDUINO\_Baud = 9600

ARDUINO\_Parity = N

ARDUINO\_Data = 8

ARDUINO\_Stop = 1

arduino\_setup()

Sleep, 1000

;First, check for a pause command

if (password = "p")

{

Pause, Toggle

break

}

if (password = "d")

{

; Use newfangled serial connection

Transform, Serials, ASC, U

arduino\_send(Serials)

Sleep, 2000

Run, audio\Chris.mp3, , Min UseErrorLevel

}

if (password = ";00000000=00007?")

{

; Use newfangled serial connection

Transform, Serials, ASC, U

arduino\_send(Serials)

Sleep, 2000

Run, audio\Chris.mp3, , Min UseErrorLevel

}

if (password = ";00000000=00007?")

{

; Use newfangled serial connection

Transform, Serials, ASC, U

arduino\_send(Serials)

Sleep, 2000

Run, audio\Tyler.mp3, , Min UseErrorLevel

}

if (password = ";00000000=00007?")

{

; Use newfangled serial connection

Transform, Serials, ASC, U

arduino\_send(Serials)

Sleep, 2000

Run, audio\Luc.mp3, , Min UseErrorLevel

}

if (password = ";00000000=00007?")

{

; Use newfangled serial connection

Transform, Serials, ASC, U

arduino\_send(Serials)

Sleep, 2000

Run, audio\Andrew.mp3, , Min UseErrorLevel

}

}

;If the string was not properly captured, play an error sound

else

{

playSound("error")

}

arduino\_close() ; this is important!! or else the program does not end when closed

}

}

;

; Extra Functions

;

OnSerialData(SerialData)

{

global serialHolder

if (InStr(SerialData, "[Start]"))

{

serialHolder := ""

; FileAppend, SSS`n, serial.txt

}

if (InStr(SerialData, "End"))

{

update = 1

; FileAppend, %SerialData% `n -- `n, serial.txt

}

serialHolder .= SerialData

if (update = 1)

{

GuiControl,, Serials, % serialHolder

serialHolder =

}

; Use newfangled serial connection

Transform, Serials, ASC, D

arduino\_send(Serials)

; FileAppend, %serialHolder% `n`n, serial.txt

}

^r::

Reload

Return

^p::

Pause, Toggle

Reload

Return

SubShutdown:

; make sure to cleanly shut down serial port on exit

arduino\_close() ; this is important!! or else the program does not end when closed

ExitApp ; shuts down the script

;

; Includes

;

#include %A\_ScriptDir%\include\Arduino.ahk

Unused code – C++

//==============================

// SuiteLock Server

// By: Chris Bero

// Last Edited:

// 11-7-2012

//==============================

//////////////////////////

// Includes

////////////////////////

#include <stdlib.h>

#include <iostream>

#include <string>

#ifdef \_WIN32

#include <Windows.h>

#else

#include <unistd.h>

#endif

#include "rs232.h"

using namespace std;

#ifdef \_WIN32

#include "ccc.h"

#else

void clslow () {

cout << string(50, '\n');

}

#endif

//////////////////////////

// Main Function

////////////////////////

int main()

{

int i = 0;

int n;

int restart = 0;

int flatline = 0;

int cport\_nr = 3; // /dev/ttyS5 (COM6 on Windows)

int bdrate = 9600; // 9600 baud

unsigned char buf[4096]; // Holds serial buffer

unsigned char charHolder[4096]; // Holds characters from buffer

string status;

string inputs;

// Open the COMport

if(OpenComport(cport\_nr, bdrate))

{

cout << "Can not open comport\n";

CloseComport(cport\_nr);

cout << "\nEnter any value to exit >\_";

cin >> i;

return(0);

}

while(1)

{

if (restart == 1)

{

if(OpenComport(cport\_nr, bdrate))

{

cout << "Can not open comport\n";

CloseComport(cport\_nr);

cout << "\nEnter any value to exit >\_";

cin >> i;

return(0);

}

else

{

restart = 0;

}

}

clslow();

if (SendByte(cport\_nr, 83)) // Send 'S' to prompt for Status Report

{

printf("\n\nSending data didn't work. \n\n");

}

else

{

// cout << "\nSent [S]\n";

}

#ifdef \_WIN32

Sleep(15); /\* it's ugly to use a sleeptimer, in a real program, change the while-loop into a (interrupt) timerroutine \*/

#else

usleep(15000); /\* sleep for 15 milliSeconds \*/

#endif

// Query the serial connection for data

n = PollComport(cport\_nr, buf, 4095);

if(n > 0)

{

buf[n] = 0; /\* always put a "null" at the end of a string! \*/

/\*

Eventually I plan for this program to pull data out of the status report

For now it's just chillin here.

\*/

for (int i = 0; i <= n; i++)

{

if (buf[i] == '$')

{

//cout << "First '$' found.";

for (int c = (i+1), d = 0; c <= n; c++, d++)

{

charHolder[d] = buf[c];

}

break;

}

}

status = (char \*)charHolder; // I'm playing around at this poing, it's true.

cout << endl << (char \*)charHolder << endl;

}

else

{

if (flatline < 5)

{

flatline ++;

}

else

{

cout << "\nNo Data recieved. Restarting Connection...\n\n";

CloseComport(cport\_nr);

restart = 1;

flatline = 0;

break;

}

}

cout << "Menu: \n[1] Refresh [2/e] Exit \n>\_";

cin >> inputs;

// I plan to replace this with a text file soon

if (inputs.compare(";025151739=02207?") == 0)

{

cout << "\nIt's Chris!";

if (SendByte(cport\_nr, 85))

{

printf("\n\nSending data didn't work. \n\n");

system("pause");

}

else

{

cout << "\nSent [U]\n";

}

}

if (inputs.compare(";025169535=01207?") == 0)

{

cout << "\nIt's Tyler!";

if (SendByte(cport\_nr, 85))

{

printf("\n\nSending data didn't work. \n\n");

system("pause");

}

else

{

cout << "\nSent [U]\n";

}

}

if (inputs.compare(";025152819=03207?") == 0)

{

cout << "\nIt's Luc!";

if (SendByte(cport\_nr, 85))

{

printf("\n\nSending data didn't work. \n\n");

system("pause");

}

else

{

cout << "\nSent [U]\n";

}

}

if (inputs.compare(";025153622=01207?") == 0)

{

cout << "\nIt's Andrew!";

if (SendByte(cport\_nr, 85))

{

printf("\n\nSending data didn't work. \n\n");

system("pause");

}

else

{

cout << "\nSent [U]\n";

}

}

if (inputs.compare("sunday") == 0)

{

cout << "\nIt's a burglar!";

if (SendByte(cport\_nr, 85))

{

printf("\n\nSending data didn't work. \n\n");

system("pause");

}

else

{

cout << "\nSent [U]\n";

}

}

if (inputs.compare("e") == 0)

{

cout << "\nExiting.";

CloseComport(cport\_nr);

return(0);

}

if (inputs.compare("1") == 0)

{

cout << "\nRefreshing.";

}

#ifdef \_WIN32

Sleep(200); /\* it's ugly to use a sleeptimer, in a real program, change the while-loop into a (interrupt) timerroutine \*/

#else

usleep(200000); /\* sleep for 200 milliSeconds \*/

#endif

}

CloseComport(cport\_nr);

return(0);

}

Future Improvements:

* Full enclosure for laptop motherboard and components
* Battery backup system
* Small android/iphone program to unlock the door from the outside
* Removal of keypad – a roommate loaned out his code
* Integration of status LEDs to place behind the fisheye lens in the door
* RFID authentication

Unfortunately, one of my roommates has gone from helping with project design ideas to actively disassembling the door lock while I’m in class. I do not believe I’ll be able to keep using the door lock as I had hoped next semester, and as such these improvements will not be implemented.

Sources:

Watchdog Timer:

<http://tushev.org/articles/electronics/48-arduino-and-watchdog-timer>

AutoHotKey Serial Connection:

<http://www.autohotkey.com/community/viewtopic.php?t=69598&start=0&postdays=0&postorder=asc&highlight=arduino>

Interfacing with AutoHotKey:

<http://arduino.cc/forum/index.php/topic,55429.0.html>

RS-232 for Linux:

<http://www.teuniz.net/RS-232/>

Converting Serial:

<http://stackoverflow.com/questions/5697047/convert-serial-read-into-a-useable-string-using-arduino>

Serial functions:

<http://www.arduino.cc/en/Serial/Print>