**Day 20 Notes**

**Creative Day 4**

Example Circuit

My Circuit

Thoughts:

* Make the Pass code user entered rather than hard coded.
* Require changing on first startup
* Store new passcode in EEPROM
* Do I have enough pins to add the 4x4 keypad to the Day 19 Circuit?
  + If so, lets wire that up and use it for the passcode. Function like final Creative Day code where the ascent works when system is unlocked.
  + Else lets use the rotary encoder and the 7 segment display for passcode work, adding push to enter.

Got it—let’s tackle this step-by-step!

### Goals

1. \*\*User-entered Passcode\*\*: The passcode will be user-defined rather than hardcoded, requiring an initial setup on the first startup.

2. \*\*EEPROM Storage\*\*: The passcode should be stored in EEPROM so it persists across power cycles.

3. \*\*Circuit Decision\*\*: Let’s see if you have enough pins for the 4x4 keypad alongside the Day 19 circuit. If not, we’ll use the rotary encoder and 7-segment display as alternatives.

### Step 1: Check Pin Availability for 4x4 Keypad

For the Day 19 circuit, I’ll need to know which components you’re currently using and their pin assignments. However, I can give you a general strategy for determining pin availability:

- \*\*Keypad\*\*: A 4x4 keypad requires 8 pins (4 for rows and 4 for columns).

- \*\*Day 19 Components\*\*: Let me know which components are in use and on which pins. I can suggest alternative pin configurations if you’re short on pins.

If we can fit the keypad into your available pins, we’ll proceed with the wiring and code. If not, we’ll switch to using the \*\*rotary encoder and 7-segment display\*\*.

Could you provide a quick rundown of your Day 19 pin usage? Once we confirm the setup, I’ll guide you through the wiring and code for the passcode entry!

Components and Pin Usage:

Rotaty Encoder

Signal (S) -- Pin 10

Ground (-) -- Common Ground Bus

7 Segment Display

CLK -- Pin 6

DIO -- Pin 5

VCC -- Common +5v Bus

GND -- Common Ground Bus

Buzzer

CLK -- Pin 2

DT -- Pin 3

Power (+) -- Common +5v Bus

GND -- Common Ground Bus

Thanks for the component and pin layout! Based on your current setup, here’s the pin usage analysis for adding the 4x4 keypad:

- \*\*4x4 Keypad\*\*: Requires 8 pins (4 rows + 4 columns).

- \*\*Available Pins\*\*: Your setup is already using \*\*Pins 2, 3, 5, 6, and 10\*\* for the rotary encoder, 7-segment display, and buzzer.

This leaves \*\*Pins 7, 8, 9, 11, 12, and 13\*\* free, which is enough to support the keypad!

Here’s the plan:

### Step 2: Wiring the 4x4 Keypad

Connect the keypad as follows (adjust if your HERO board uses different pin labels for these):

- \*\*Keypad Row Pins (R1-R4)\*\* → Arduino Pins \*\*7, 8, 9, 11\*\*

- \*\*Keypad Column Pins (C1-C4)\*\* → Arduino Pins \*\*12, 13, A0, A1\*\*

With this setup, all components fit, and we have enough pins for the keypad. The next steps involve modifying the code to:

1. Set up and check the passcode.

2. Store and retrieve the passcode from EEPROM.

3. Use the keypad to enter the passcode for system access.

### Step 3: Code for Passcode Setup and Entry

Here’s an outline for the updated code structure:

1. \*\*Passcode Setup and Validation\*\*: Prompts the user to set up a new passcode on the first startup. Stores the new passcode in EEPROM.

2. \*\*Passcode Entry\*\*: After the first startup, retrieves the passcode from EEPROM and requires the user to unlock the system with it.

Would you like the code to proceed with this setup?

#include <Keypad.h>

#include <EEPROM.h>

// Constants for minimum and maximum PIN length

const int minPinLength = 4;

const int maxPinLength = 8;

const int maxAttempts = 5;

const int eepromStartAddress = 0; // EEPROM address for storing PIN

// Keypad setup

const char BUTTONS[4][4] = {

{ '1', '2', '3', 'A' },

{ '4', '5', '6', 'B' },

{ '7', '8', '9', 'C' },

{ '\*', '0', '#', 'D' }

};

const byte ROW\_PINS[4] = { 7, 8, 9, 11 };

const byte COL\_PINS[4] = { 12, 13, A0, A1 };

Keypad heroKeypad = Keypad(makeKeymap(BUTTONS), ROW\_PINS, COL\_PINS, 4, 4);

char currentPIN[maxPinLength + 1]; // Stores the active PIN

int attemptCount = 0;

bool isLocked = true; // System starts in locked state

void setup() {

Serial.begin(9600);

// Load the PIN from EEPROM on startup

if (loadPINFromEEPROM()) {

Serial.println("Stored PIN loaded. Enter PIN to unlock.");

} else {

Serial.println("No PIN found. Set a new PIN (4-8 chars, end with #):");

setupNewPIN();

}

}

void loop() {

if (isLocked) {

Serial.println("System is locked. Enter PIN to unlock:");

if (validatePIN()) {

isLocked = false;

Serial.println("System Unlocked! Options: '#' to lock, '\*' to change PIN, other keys to work.");

} else {

attemptCount++;

if (attemptCount >= maxAttempts) {

Serial.println("Too many failed attempts. System locked.");

while (true); // Lock system permanently

}

}

} else {

char button\_character = heroKeypad.getKey();

if (button\_character) {

if (button\_character == '#') { // Lock system

isLocked = true;

Serial.println("System locked. Enter PIN to unlock.");

} else if (button\_character == '\*') { // Change PIN

Serial.println("Enter current PIN to change:");

if (validatePIN()) {

Serial.println("PIN accepted. Set a new PIN:");

setupNewPIN();

Serial.println("New PIN set. System unlocked.");

} else {

Serial.println("Incorrect PIN. Returning to unlocked state.");

}

} else { // Any other character to "work"

Serial.print("Working... You pressed: ");

Serial.println(button\_character);

}

}

}

}

// Setup a new PIN with verification

void setupNewPIN() {

char tempPIN[maxPinLength + 1];

Serial.println("Enter new PIN:");

if (getPINInput(tempPIN)) {

Serial.println("Re-enter new PIN to confirm:");

char verifyPIN[maxPinLength + 1];

if (getPINInput(verifyPIN) && strcmp(tempPIN, verifyPIN) == 0) {

strcpy(currentPIN, tempPIN);

savePINToEEPROM();

Serial.println("New PIN set successfully!");

} else {

Serial.println("PINs did not match. Restart setup.");

setupNewPIN();

}

}

}

// Save the current PIN to EEPROM

void savePINToEEPROM() {

int length = strlen(currentPIN);

EEPROM.write(eepromStartAddress, length); // Store the length

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

EEPROM.write(eepromStartAddress + 1 + i, currentPIN[i]);

}

}

// Load the PIN from EEPROM

bool loadPINFromEEPROM() {

int length = EEPROM.read(eepromStartAddress);

if (length < minPinLength || length > maxPinLength) {

return false;

}

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

currentPIN[i] = EEPROM.read(eepromStartAddress + 1 + i);

}

currentPIN[length] = '\0';

return true;

}

// Get PIN input with "#" to end entry

bool getPINInput(char\* pinBuffer) {

int charIndex = 0;

while (true) {

char key = heroKeypad.getKey();

if (key) {

if (key == '#') {

pinBuffer[charIndex] = '\0';

if (charIndex >= minPinLength && charIndex <= maxPinLength) {

return true;

} else {

Serial.println("Invalid PIN length. Try again.");

return false;

}

}

if (charIndex < maxPinLength) {

pinBuffer[charIndex] = key;

Serial.print("\*");

charIndex++;

}

}

}

}

// Validate entered PIN against currentPIN

bool validatePIN() {

char enteredPIN[maxPinLength + 1];

if (getPINInput(enteredPIN)) {

if (strcmp(enteredPIN, currentPIN) == 0) {

Serial.println("Access Granted!");

return true;

}

}

Serial.println("Incorrect PIN.");

return false;

}