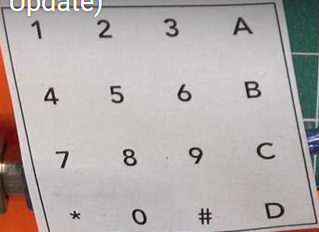
**Day 15 Notes**

Example Circuit

My Circuit



Preliminary Thoughts:

* Change Code to only give valid or not valid after all PIN characters are entered as suggested in the Day 15 Lesson video.
* Add required entry of a NEW PIN at startup, before entering Loop.
* Require a second verifying entry of the PIN before completing a change.
* Allow the PIN to be from 4 to 8 characters long. Use the “#” key to end entry of a new PIN.
* Can we make user entered Data (New PIN) persist past power off / on cycle? If so, keep the New Pin after restart and don’t require another NEW PIN on restart.

Thank you for catching that! Let’s update the code to address both requirements:

1. \*\*User-Defined Starting PIN\*\*: Prompt the user to set up a new PIN from the keypad at the start, rather than using a hard-coded PIN.

2. \*\*Variable PIN Length\*\*: Allow the PIN to range from 4 to 8 characters, finalized by the `#` key.

Here's the revised code:

```cpp

// Define maximum and minimum PIN length

const int minPinLength = 4;

const int maxPinLength = 8;

const int maxAttempts = 5;

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

int attemptCount = 0; // Track number of attempts

void setup() {

Serial.begin(9600); // Start serial communication

pinMode(LED\_BUILTIN, OUTPUT); // Set up onboard LED for feedback

// Prompt for initial user-defined PIN setup

Serial.println("Set a new PIN (4-8 characters). End with #:");

setupNewPIN();

}

void loop() {

if (attemptCount >= maxAttempts) {

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

while (true); // Lock the system

}

Serial.println("Enter PIN:");

if (validatePIN()) {

Serial.println("Access Granted!");

digitalWrite(LED\_BUILTIN, HIGH); // Turn on LED for valid entry

delay(1000); // Hold success for visibility

} else {

attemptCount++;

Serial.print("Incorrect PIN. Attempts remaining: ");

Serial.println(maxAttempts - attemptCount);

digitalWrite(LED\_BUILTIN, LOW); // Turn off LED for invalid entry

delay(1000);

}

}

// Set up a new PIN with verification

void setupNewPIN() {

char tempPIN[maxPinLength + 1]; // Temporary storage for first PIN entry

// First entry of new PIN

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

if (getPINInput(tempPIN)) {

Serial.println("Re-enter new PIN for verification:");

// Second entry to confirm

char verifyPIN[maxPinLength + 1];

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

strcpy(currentPIN, tempPIN); // Set the new PIN

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

} else {

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

setupNewPIN(); // Retry setup if verification fails

}

}

}

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

bool getPINInput(char\* pinBuffer) {

int charIndex = 0;

while (true) {

if (Serial.available()) {

char inputChar = Serial.read();

// Check for end of entry with "#"

if (inputChar == '#') {

pinBuffer[charIndex] = '\0'; // End the PIN string

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

return true;

} else {

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

return false; // Invalid length

}

}

// Add character if within limits

if (charIndex < maxPinLength) {

pinBuffer[charIndex] = inputChar;

Serial.print("\*"); // Mask input with \*

charIndex++;

}

}

}

}

// Validate entered PIN against currentPIN

bool validatePIN() {

char enteredPIN[maxPinLength + 1];

if (getPINInput(enteredPIN)) {

return strcmp(enteredPIN, currentPIN) == 0;

}

return false;

}

```

### Key Changes:

- \*\*User-defined Starting PIN\*\*: The `setupNewPIN()` function now requires a user-defined PIN at startup. This process involves two entries—the first for creating the PIN and the second for verification.

- \*\*Variable PIN Length\*\*: `getPINInput()` checks that the entered PIN meets the 4–8 character requirement. The `#` key finalizes the input, making it possible to vary the PIN length.

### How It Works:

- \*\*Set a New PIN\*\*: At startup, `setupNewPIN()` is called to prompt the user to enter and confirm a new PIN.

- \*\*Variable Length\*\*: The user-defined PIN can be between 4 and 8 characters, and the `#` key signifies the end of input.

- \*\*Attempt Limit\*\*: The program limits attempts to 5. After 5 incorrect attempts, the system locks.

This should now fully meet your requirements for a customizable PIN setup and validation process! Let me know if you need further customization or if you’d like to add any new features.