TERRAN BLAKE // WEDNESDAY 7:30 // ECE 241

```
#include <LiquidCrystal.h>
LiquidCrystal LcdDriver(12, 11, 5, 6, 7, 8);

//variables for the pass
char passKey1 = 80;
char passKey2 = 65;
char passKey3 = 83;
char passKey4 = 83;

//Holds the value of inputs
char passInput1 = constrain(65, 65, 90);    //probably needs changed to x for input
char passInput2 = constrain(65, 65, 90);    //probably needs to go inside states
char passInput3 = constrain(65, 65, 90);
char passInput4 = constrain(65, 65, 90);

boolean buttonpress = LOW; //stores the value of the button press

enum PinAssignments {
  encoderPinA = 2,  //encoder turns on pin 2 and 3
  encoderPinB = 3,
  encoderPinC = 4,  //encoder press on pin 4
};

//keeps track of the position of the encoder
unsigned int encoderPos = 65;
unsigned int lastReportedPos = 1;

//values for the interupt monitor
boolean A_set = false;
boolean B_set = false;

enum PassStates{ digit1, digit2, digit3, digit4};

PassStates currentState;
void NextState()
{
  switch(currentState) {

    case digit1:
      passInput1 = encoderPos;
      if(digitalRead(4) == LOW) {
        delay(500);
        currentState = digit2;
      }
      break;
```

```
   case digit2:
  passInput2 = encoderPos;

  if(digitalRead(4) == LOW) {
       delay(500);
       currentState = digit3;
       }
     break;

   case digit3:
  passInput3 = encoderPos;
  if(digitalRead(4) == LOW) {
       delay(500);
       currentState = digit4;
    }
     break;

   case digit4:
  passInput4 = encoderPos;
  if(digitalRead(4) == LOW) {
       currentState = digit1;
     }
     break;
 }
}

void setup() {
//Lcd setup
 LcdDriver.begin(16,2);
 LcdDriver.clear();
 LcdDriver.setCursor(0,0);

 Serial.begin(9600);

//Pin Assignments
 pinMode(encoderPinA, INPUT);
 pinMode(encoderPinB, INPUT);
 pinMode(encoderPinC, INPUT);
 digitalWrite(encoderPinA, HIGH);  // turn on pullup resistor
 digitalWrite(encoderPinB, HIGH);  // turn on pullup resistor
 digitalWrite(encoderPinC, HIGH);

// encoder pin on interrupt 0 (pin 2)
 attachInterrupt(0, MonitorA, CHANGE);
// encoder pin on interrupt 1 (pin 3)
 attachInterrupt(1, MonitorB, CHANGE);
}
```

```
void loop(){
  NextState();

  //Keeps the encoder position within the limits of the ASCII password
  if (encoderPos > 90) {
    encoderPos = 65;
  }

  if (encoderPos < 66) {
    encoderPos = 90;
  }

  if (lastReportedPos != encoderPos) {
    Serial.print(encoderPos, DEC);
    Serial.println();
    lastReportedPos = encoderPos;
  }

//changes values for passInput
  //if(encoderPos + 4 >){   //encoder is rotated change value
  //  passInput1++;
   if(passInput2 > 89){
     passInput2 = 65;
  }
  //}
  buttonpress = digitalRead(encoderPinC); //reads presses

  if(buttonpress = LOW){
            //switch states
  }

//LCD passkey is correct or incorrect
  LcdDriver.setCursor(0,0);
  LcdDriver.print(passInput1);
  LcdDriver.print(passInput2);
  LcdDriver.print(passInput3);
  LcdDriver.print(passInput4);

  if (passInput1 == passKey1 && passInput2 == passKey2 && passInput3 == passKey3 && passInput4 ==
passKey4) {
    LcdDriver.setCursor(0,1);
    LcdDriver.print("unlocked");
  }
  else{
    LcdDriver.setCursor(0,1);
    LcdDriver.print("locked");
  }
```

```
}

// Interrupt on A changing state
void MonitorA(){
  // Test transition
  A_set = digitalRead(encoderPinA) == HIGH;
  // and adjust counter + if A leads B
  encoderPos += (A_set != B_set) ? +1 : -1;
}

// Interrupt on B changing state
void MonitorB(){
  // Test transition
  B_set = digitalRead(encoderPinB) == HIGH;
  // and adjust counter + if B follows A
  encoderPos += (A_set == B_set) ? +1 : -1;
}
```