

Post date:

Tuesday, May 3, 2016 - 23:54

Last modified:

Wednesday, May 4, 2016 - 00:22

Heres the arduino code code for the [Pneumatic-air-piston-for-table-saw-sled-controlled-by-an-arduino](#).

[Download Link:](#)

The code is a bit of a mess, after everything was working i never whent back to clean it up, sorry.

```
#include <digitalwritefast.h>
#include <wire.h>
#include <adafruit_gfx.h>
#include <gfont.h>
#include <adafruit_ledbackpack.h>
#include <keypad.h>

const int pin7SegmentDisplaySDA = 20; // SDA GREEN data Analog-4 or Digital 20 for the Mega
const int pin7SegmentDisplaySCL = 21; // SCL WHITE clock Analog-5 or Digital 21 for the Mega
Adafruit_7segment segmentCounter = Adafruit_7segment();

const byte ROWS = 4; // Four rows
const byte COLS = 4; // Three columns
// Define the Keymap
char keys[ROWS][COLS] = {
  {'1','2','3', 'A'},
  {'4','5','6', 'B'},
  {'7','8','9', 'C'},
  {'*','0','#', 'D'}
};
// Connect keypad ROW0, ROW1, ROW2 and ROW3 to these Arduino pins.
// 36 = yellow
// 37 = green
byte rowPins[ROWS] = { 30, 32, 34, 36 };
// Connect keypad COL0, COL1 and COL2 to these Arduino pins.
byte colPins[COLS] = { 31, 33, 35, 37};

// Create the Keypad
Keypad kpd = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

const int pinPiston_Out_Switch = 5; // yellow "HOME" 980X
const int pinPiston_In_Switch = 6; // orange 987X

const int pinPiston_Out_Relay = 3; // yellow
const int pinPiston_In_Relay = 4; // orange

const int pinSwitchCountUp = 10; // yellow
const int pinSwitchCountDown = 11; // blue
volatile char countUPorDOWN = 'u';
```

```
const int pinClearCounter = 8; // white // reset button

const int pinHome = 9; // blue
//const int pinReset = 12; // on the board with 1k restore
const int pinStart = 2; // yellow

const int LEDrunning = 7; // green
const int LEDreset = 52; // white
const int LEDblinkHome = 53; // greeb
const int LEDblinkERROR = 50; // BLUE

//const int LEDreset = 50; // BLUE

// reverse
const int manualOut = 40; // blue
const int LEDmanualOut = 41; // yellow

// Forward
const int manualIn = 42; // green
const int LEDmanualIn = 43; // white

volatile int pistonOut = 0;
volatile int pistonIn = 0;

volatile int triggerError = 0;
volatile int runDone = 1;

volatile int partCounter = 0;
int partCounterSetTo = 0;
char partCounterSetToArray[5];
int partCounterSetToArrayIndex=0;
int settingCounter = 0;
char keyPressed = 0;

volatile int cycleRunning = 0;

volatile int isHome = 0;

unsigned long blink_PreviousMillis = 0;
const long blink_Interval = 500;
int blink_ledState = LOW;

const int pinBuzz = 12; // orange
int buzzOn = 1;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);

  segmentCounter.begin(0x70);
  zeroDisplay();

  pinMode(LEDrunning, OUTPUT);
  pinMode(LEDreset, OUTPUT);
  pinMode(LEDblinkHome, OUTPUT);
  pinMode(LEDblinkERROR, OUTPUT);
```

```

pinMode(LEDmanualOut, OUTPUT);
pinMode(LEDmanualIn, OUTPUT);

pinMode(pinPiston_Out_Switch, INPUT_PULLUP);
digitalWriteFast(pinPiston_Out_Switch, HIGH);

pinMode(pinPiston_In_Switch, INPUT_PULLUP);
digitalWriteFast(pinPiston_In_Switch, HIGH);

pinMode(pinPiston_Out_Relay, OUTPUT);
digitalWriteFast(pinPiston_Out_Relay, 0);

pinMode(pinPiston_In_Relay, OUTPUT);
digitalWriteFast(pinPiston_In_Relay, 0);

pinMode(pinStart, INPUT_PULLUP);

pinMode(pinHome, INPUT_PULLUP);
//pinMode(pinReset, OUTPUT);
pinMode(pinClearCounter, INPUT_PULLUP);

pinMode(pinSwitchCountUp, INPUT_PULLUP);
pinMode(pinSwitchCountDown, INPUT_PULLUP);

pinMode(manualOut, INPUT_PULLUP);
pinMode(manualIn, INPUT_PULLUP);

pinMode(pinBuzz, OUTPUT);
}

```

```

void(* resetFunc) (void) = 0; //declare reset function @ address 0

```

```

void loop() {

    //Serial.print("pinPiston_Out_Relay");
    //Serial.println(digitalReadFast(pinPiston_Out_Relay));

    //Serial.print("pinPiston_In_Relay");
    //Serial.println(digitalReadFast(pinPiston_In_Relay));

    //digitalWriteFast(pinPiston_In_Relay, LOW);
    //digitalWriteFast(pinPiston_Out_Relay, LOW);

    if (digitalReadFast(pinPiston_Out_Switch) == 0){
        isHome = 1;
        digitalWriteFast(LEDblinkHome, HIGH);
    } else {

        isHome = 0;

        if(cycleRunning == 0){
            char ledArray[2];
            ledArray[1] = LEDblinkHome;
            makeLEDBlink(ledArray);
        }
    }
}

```

```

    } else {
        digitalWriteFast(LEDblinkHome, LOW);
    }
}

// Make sure values are not open when they should not be...
if (digitalReadFast(pinPiston_Out_Switch) == 0){
    digitalWriteFast(pinPiston_Out_Relay, LOW);
}
if (digitalReadFast(pinPiston_In_Switch) == 0){
    digitalWriteFast(pinPiston_In_Relay, LOW);
}

// LED RELAY ACTIVE TRIGGER
if (digitalReadFast(pinPiston_Out_Relay) == 1){
    ledOnOff(true, LEDmanualOut);
} else {
    ledOnOff(false, LEDmanualOut);
}
if (digitalReadFast(pinPiston_In_Relay) == 1){
    ledOnOff(true, LEDmanualIn);
} else {
    ledOnOff(false, LEDmanualIn);
}

// Manual OUT
if (digitalReadFast(manualOut)

```

**LOW && digitalReadFast(pinPiston\_Out\_Switch)**

```

HIGH && cycleRunning == 0){
    cycleRunning = 0;
    extentPiston();
} else if (digitalReadFast(manualOut)

```

**HIGH && cycleRunning**

```

0){
    digitalWriteFast(pinPiston_Out_Relay, LOW);
}

```

**LOW && digitalReadFast(pinPiston\_In\_Switch)**

```

HIGH && cycleRunning == 0){
    cycleRunning = 0;
    retractPiston();
} else if (digitalReadFast(manualIn)

```

**HIGH && cycleRunning**

```

0){
    digitalWriteFast(pinPiston_In_Relay, LOW);
}

```

**LOW && isHome**

```

1 && cycleRunning
0 && triggerError

```

```

0){

```

```
if (digitalReadFast(pinStart)
```

## **LOW && isHome**

```
1 && cycleRunning == 0){
```

```
//if (digitalReadFast(pinStart)
```

## **LOW && cycleRunning**

```
0){
```

```
    delay(5);  
    triggerError = 0;  
    isHome = 0;  
    cycleRunning = 0;  
    //sendHome();  
    //stop();  
    startCycle();  
    delay(30);  
    runDone = 0;  
    cycleRunning = 1;  
} else if(digitalReadFast(pinStart)
```

## **LOW && isHome**

```
0 && cycleRunning == 0){
```

```
    ledOnOff(true, LEDblinkERROR);
```

```
    } else {
```

```
        if(triggerError == 0){  
            ledOnOff(false, LEDblinkERROR);  
        } else {  
            //ledOnOff(false, LEDblinkERROR);  
        }  
        //ledOnOff(false, LEDblinkERROR);
```

```
    }
```

```
    // END OF CYCLE
```

```
if (digitalReadFast(pinPiston_Out_Switch)
```

## **LOW && digitalReadFast(pinPiston\_In\_Relay)**

```
LOW && cycleRunning == 1){
```

```
//if (digitalReadFast(pinPiston_Out_Switch)
```

## **LOW && cycleRunning**

```
1){
```

```
    //isHome = 1;  
    partCounter = count(partCounter);  
    displayCount(partCounter);  
    stop();  
    cycleRunning = 0;
```

```
    }
```

```
    // AUTO EXTENT PISTION WHILE IN CYCLE
```

```
if (digitalReadFast(pinPiston_In_Switch)
```

## **LOW && cycleRunning**

```
1){
```

```
    extentPiston();
```

```
    }
```

```
    // COUNTER SWTICH
```

```
if (digitalReadFast(pinSwitchCountUp) == LOW){
```

```

        countUPorDOWN = 'd';
    } else if (digitalReadFast(pinSwitchCountDown) == LOW ){
        countUPorDOWN = 'u';
    } else {
        countUPorDOWN = 'n';
    }

    // HOME
    if (digitalReadFast(pinHome) == LOW){
        isHome = 0;
        cycleRunning = 0;
        sendHome();
    }

```

## 0 && countUPorDOWN

```

    if (partCounter
'd' && runDone == 0){
        segmentCounter.blinkRate(1);
        triggerError = 1;
    } else {
        triggerError = 0;
    }

```

## LOW && countUPorDOWN

```

'n'){
        //pinMode(pinReset, OUTPUT);          // sets the digital pin as output
        //digitalWriteFast(pinReset, LOW);    // sets the LED off
        ledOnOff(true, LEDreset);
        delay(5);
        //digitalWriteFast(pinReset, HIGH);
        resetFunc();
        //digitalWriteFast(pinReset, HIGH);
        //resetFunc(); //call reset
    }

```

```

    if (digitalReadFast(pinClearCounter) == LOW && countUPorDOWN != 'n'){
        zeroDisplay();
        runDone = 1;
    }

```

```

    char key = kpd.getKey();
    if(key && cycleRunning == 0) {
        //Serial.println(key);
        keyPadPressed(key);
    }

```

## LOW || (key && key

```

'C')){
        Serial.println(key);
        ledOnOff(true, LEDreset);
    } else {
        ledOnOff(false, LEDreset);
    }

```

```

}

if(settingCounter == 1) {
    segmentCounter.blinkRate(1);
    displayCount(partCounterSetTo);
} else {

```

## 0 && triggerError

```

1) {
    segmentCounter.blinkRate(1);
    displayCount(partCounter);
} else {
    segmentCounter.blinkRate(0);
    displayCount(partCounter);
}

}

```

## 0 && cycleRunning

```

//if (isHome
0){
    //char ledArray[3];
    //ledArray[0] = LEDblinkHome;
    //ledArray[1] = LEDblinkERROR;
    //makeLEDblink(ledArray);
//} else {
    //digitalWriteFast(LEDblinkHome, LOW);

    //if (triggerError == 0) {
        //digitalWriteFast(LEDblinkERROR, LOW);
    //}

//}

if (triggerError == 1){
    char ledArray[2];
    ledArray[1] = LEDblinkERROR;
    makeLEDblink(ledArray);
}

//if (isHome == 1){
    //digitalWriteFast(LEDblinkHome, LOW);
//} else {
    //digitalWriteFast(LEDblinkHome, HIGH);
//}

}

```

```

void makeLEDblink_(char led[]){

```

```

    // try and loop over the LEDs passed to we can blink more then one at a time
    for( int a = 0; a < (sizeof(led)); a = a + 1 ){
        unsigned long blink_CurrentMillis = millis();

```

```

    if (blink_CurrentMillis - blink_PreviousMillis >= blink_Interval) {
        // save the last time you blinked the LED
        blink_PreviousMillis = blink_CurrentMillis;

        // if the LED is off turn it on and vice-versa:
        if (blink_ledState == LOW) {
            blink_ledState = HIGH;
        } else {
            blink_ledState = LOW;
        }
        digitalWriteFast(led[a], blink_ledState);
    }
}
}

```

```

}

```

```

void makeLEDblink(char led[]){

```

```

    unsigned long blink_CurrentMillis = millis();
    if (blink_CurrentMillis - blink_PreviousMillis >= blink_Interval) {
        // save the last time you blinked the LED
        blink_PreviousMillis = blink_CurrentMillis;

        // if the LED is off turn it on and vice-versa:
        if (blink_ledState == LOW) {
            blink_ledState = HIGH;
        } else {
            blink_ledState = LOW;
        }

        for( int a = 0; a < (sizeof(led)); a = a + 1 ){
            digitalWriteFast(led[a], blink_ledState);
        }
    }
}

```

```

}

```

```

void keyPadPressed(char keyIs){

```

```

    switch (keyIs)
    {
        case '1':
        case '2':
        case '3':
        case '4':
        case '5':
        case '6':
        case '7':
        case '8':
        case '9':
        case '0':
            addKeyPadNumber(keyIs);
            break;

        case 'A':
            //
            break;
    }
}

```



```

    case 'B':
        //
        break;

    case 'C': // clear
        zeroDisplay();
        break;

    case 'D':
        deleteKeyPadChr();
        break;

    case '*':
        exitKeyPad();
        break;

    case '#':
        poundKey();
        break;

}

// Serial.println(keyIs);
keyPressed = keyIs;
}

void poundKey(){

    if(settingCounter == 1){
        settingCounter = 0;
        partCounter = partCounterSetTo;
        partCounterSetTo = 0;
        partCounterSetToArrayIndex = 0;
        memset(partCounterSetToArray, 0, sizeof(partCounterSetToArray));
    }

}

void exitKeyPad(){

    settingCounter = 0;
    partCounterSetTo = 0;
    partCounterSetToArrayIndex = 0;
    memset(partCounterSetToArray, 0, sizeof(partCounterSetToArray));

}

void addKeyPadNumber(char numIs){

    settingCounter = 1;

    if (partCounterSetToArrayIndex == 0){
        memset(partCounterSetToArray, 0, sizeof(partCounterSetToArray));
        partCounterSetTo = 0;
    }

    partCounterSetToArray[partCounterSetToArrayIndex] = numIs;
}

```

```
partCounterSetTo = atoi(partCounterSetToArray);
partCounterSetToArrayIndex +=1;
```

```
}
```

```
void deleteKeyPadChr(){
```

```
    //Serial.println(partCounterSetToArrayIndex, DEC);
```

```
    if (partCounterSetToArrayIndex > 0){
        settingCounter = 1;
```

```
        partCounterSetToArrayIndex -=1;
        //Serial.print("BIGGER new index count = ");
        //Serial.println(partCounterSetToArrayIndex, DEC);
```

```
        char TEMPpartCounterSetToArray[5];
        //TEMPpartCounterSetToArray = partCounterSetToArray;
        memcpy( TEMPpartCounterSetToArray, partCounterSetToArray, 5 );
```

```
        //Serial.print("partCounterSetToArray index 1 = ");
        //Serial.print(partCounterSetToArray[1]);
        //Serial.println(" ----- ");
        memset(partCounterSetToArray, 0, sizeof(partCounterSetToArray));
```

```
        for( int a = 0; a < partCounterSetToArrayIndex; a = a + 1 )
        {
```

```
            //Serial.print("a = ");
            //Serial.println(a);
            //Serial.print("tmp = ");
            //Serial.println(TEMPpartCounterSetToArray[a]);
            partCounterSetToArray[a] = TEMPpartCounterSetToArray[a];
        }
```

```
        partCounterSetTo = atoi(partCounterSetToArray);
```

```
    }
```

```
}
```

```
void sendHome(){
```

```
    //Serial.println("HOMING OUT");
```

```
    cycleRunning = 0;
    digitalWriteFast(pinPiston_In_Relay, LOW);
    //digitalWriteFast(pinPiston_Out_Relay, LOW);
```

```
    if (digitalReadFast(pinPiston_Out_Switch) == LOW){
        isHome = 1;
        digitalWriteFast(pinPiston_Out_Relay, LOW);
        ledOnOff(false, LEDmanualOut);
    }
```

```
    if (isHome == 0){
        digitalWriteFast(pinPiston_Out_Relay, HIGH);
        ledOnOff(true, LEDmanualOut);
    }
```

```
    sendHome();
}
```

```

}

void startCycle(){
    ledOnOff(true, LEDrunning);
    digitalWriteFast(pinPiston_In_Relay, LOW);
    delay(30);
    digitalWriteFast(pinPiston_Out_Relay, LOW);
    delay(30);
    retractPiston(); // cut
    delay(30);
}

void extentPiston(){
    digitalWriteFast(pinPiston_In_Relay, LOW);
    digitalWriteFast(pinPiston_Out_Relay, LOW);
    digitalWriteFast(pinPiston_Out_Relay, HIGH);
}

void retractPiston(){
    digitalWriteFast(pinPiston_In_Relay, LOW);
    digitalWriteFast(pinPiston_Out_Relay, LOW);
    digitalWriteFast(pinPiston_In_Relay, HIGH);
}

void stop(){
    digitalWriteFast(pinPiston_Out_Relay, LOW);
    digitalWriteFast(pinPiston_In_Relay, LOW);
    delay(10);
    digitalWriteFast(pinPiston_Out_Relay, LOW);
    digitalWriteFast(pinPiston_In_Relay, LOW);
    ledOnOff(false, LEDrunning);
}

char count(int num){

    if (countUPorDOWN == 'u'){
        num = (num + 1);
    }

    if (countUPorDOWN == 'd'){
        num = (num - 1);
    }

    return num;
}

void displayCount(int num){

    segmentCounter.print(num);
    segmentCounter.writeDisplay();

    if(num == 0){
        buzzard(true);
    }
}

```

```

}

void zeroDisplay(){
    partCounter = 0;
    segmentCounter.print(0000, DEC);
    segmentCounter.writeDisplay();
    buzzard(false);
}

void buzzard(bool l){

    if(l){
        digitalWriteFast(pinBuzz, HIGH);
        buzzOn = 1;
    } else {
        digitalWriteFast(pinBuzz, LOW);
        buzzOn = 0;
    }
}

void ledOnOff(bool l,int LED){

    if(l){
        digitalWriteFast(LED, HIGH);
    } else {
        digitalWriteFast(LED, LOW);
    }
}
}

```

Mobile QR Barcodes:

**Save Article as PDF**



**Current URL Address**



Mobile: File Download for - **[AirPiston.ino](#)**



## <br />Other site pages which link here:<br />

Category:

[arduino code](#)

Tags:

[Arduino](#)

-  [Save page as PDF](#)

File Downloads:

 [AirPiston.ino](#)

---