متحكمات صغرية ونظم مضمنة

Lecture No. 4

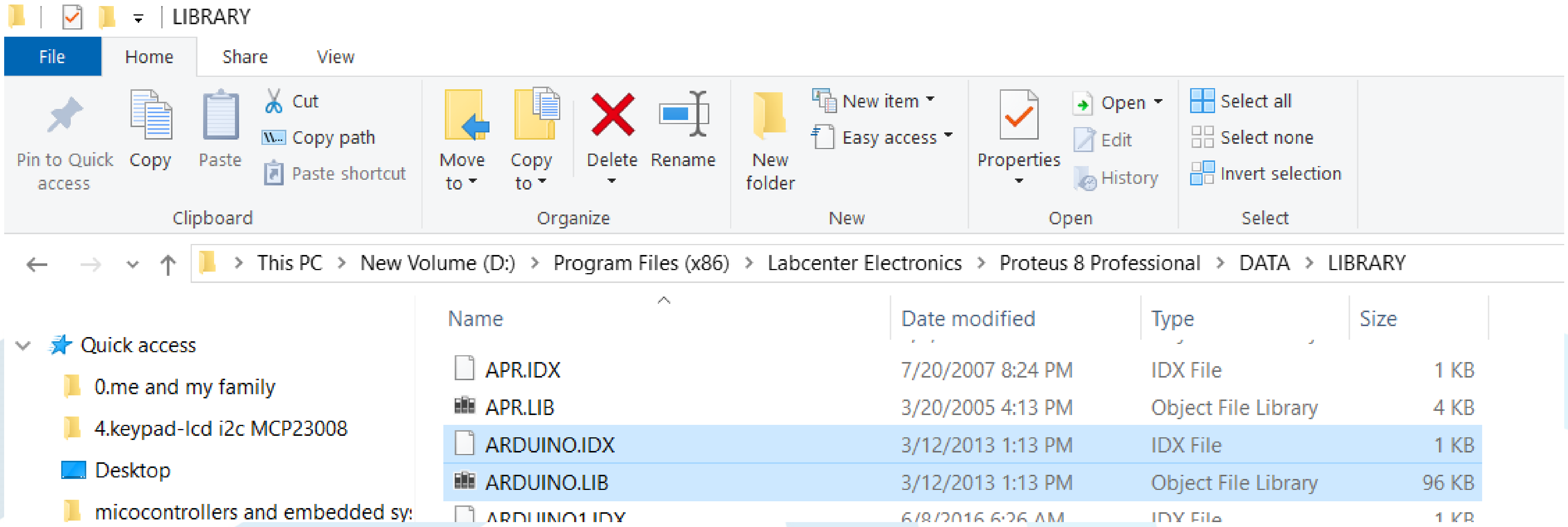
ميكاترونيكس- سنة رابعة

Dr. Eng. Essa Alghannam
Ph.D. Degree in Mechatronics Engineering

2025

Add proteus library

- D:\Program Files (x86)\Labcenter Electronics\Proteus 8 Professional\DATA\LIBRARY

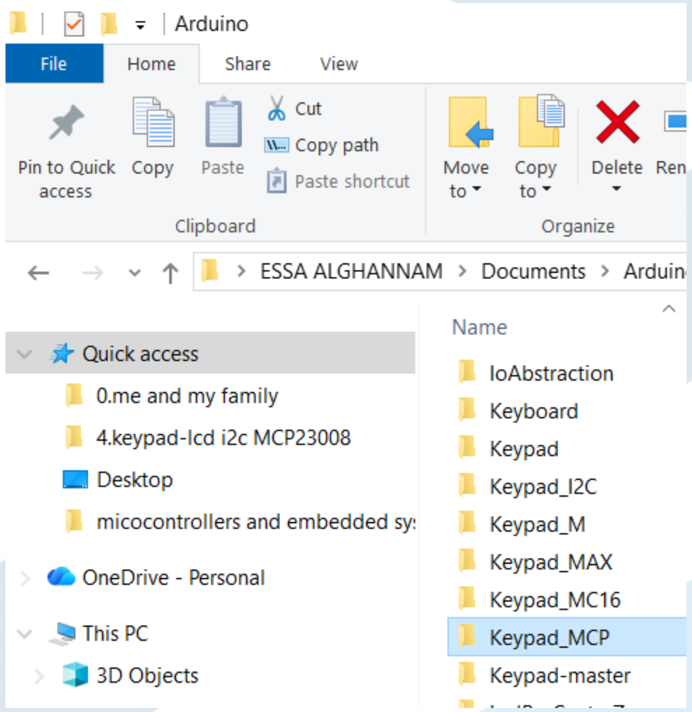


Name	Date modified	Type	Size
APR.IDX	7/20/2007 8:24 PM	IDX File	1 KB
APR.LIB	3/20/2005 4:13 PM	Object File Library	4 KB
ARDUINO.IDX	3/12/2013 1:13 PM	IDX File	1 KB
ARDUINO.LIB	3/12/2013 1:13 PM	Object File Library	96 KB
ARDUINO1.IDX	6/8/2016 6:26 AM	IDX File	1 KB

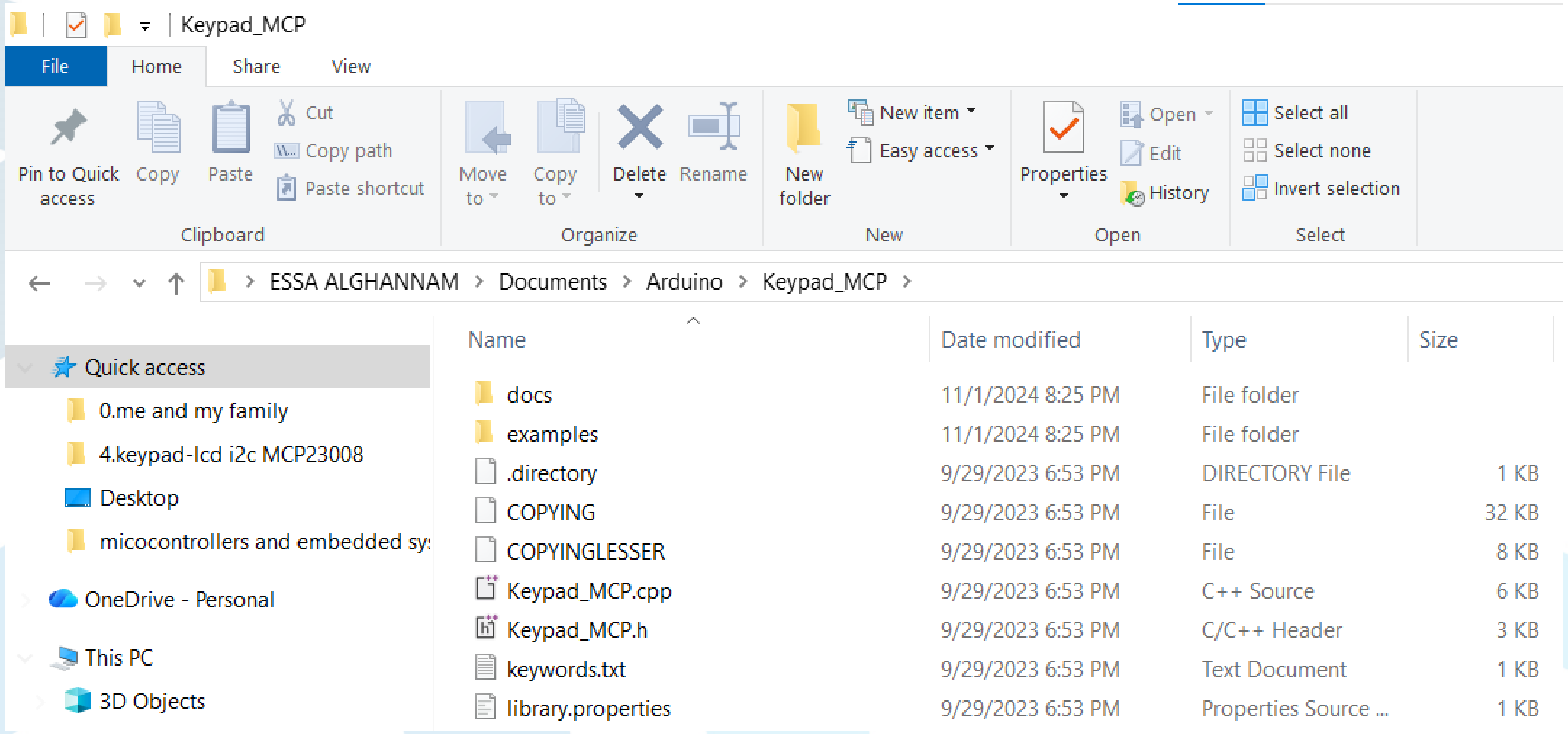
Add Arduino library

- Method 1

  C:\Users\essaa\Documents\Arduino
  C:\Users\essaa\AppData\Local\Arduino15\packages\arduino\hardware\avr\1.8.6\libraries



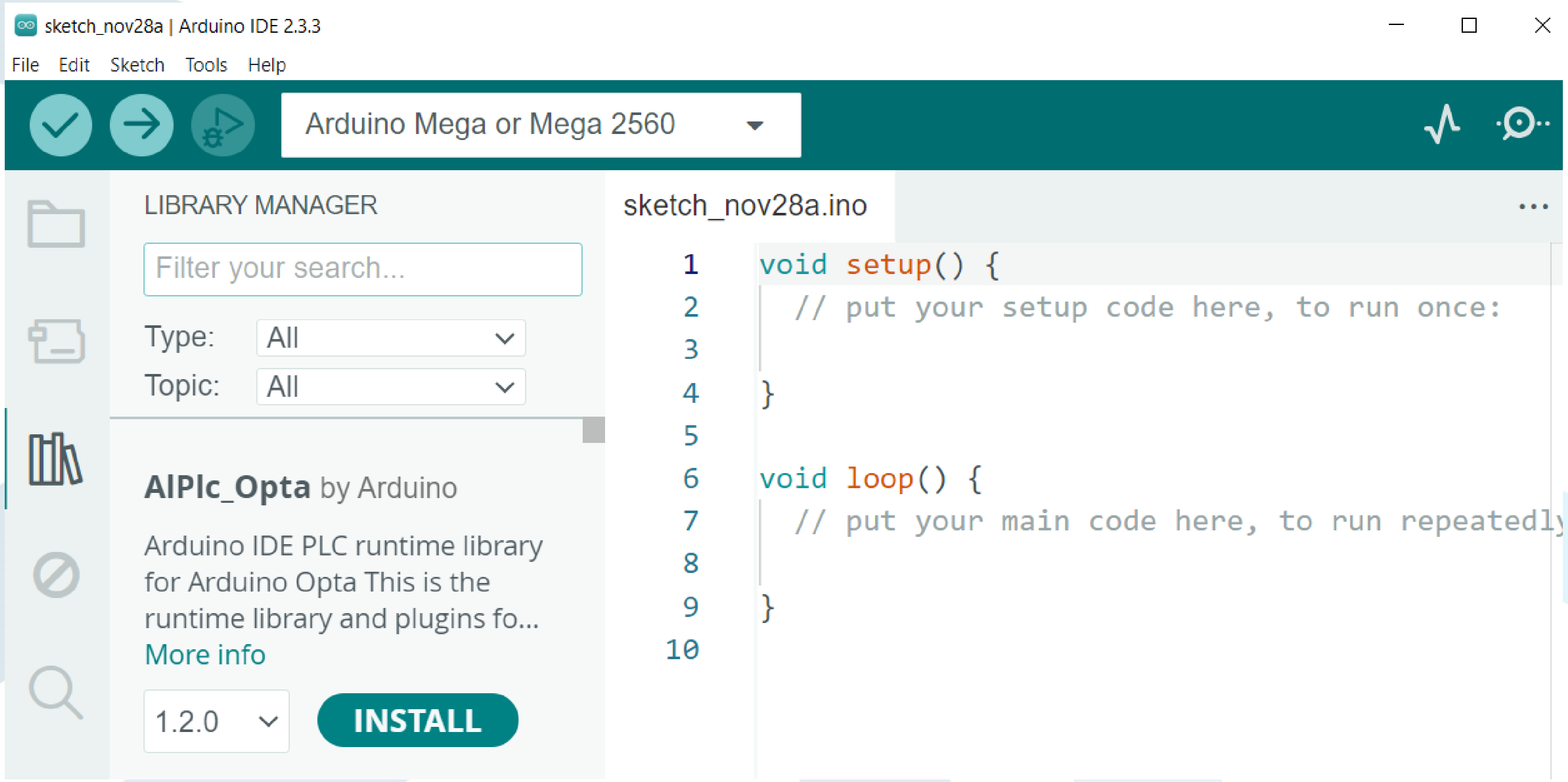
Name
IoAbstraction
Keyboard
Keypad
Keypad_I2C
Keypad_M
Keypad_MAX
Keypad_MC16
Keypad_MCP
Keypad-master



Name	Date modified	Type	Size
docs	11/1/2024 8:25 PM	File folder	
examples	11/1/2024 8:25 PM	File folder	
.directory	9/29/2023 6:53 PM	DIRECTORY File	1 KB
COPYING	9/29/2023 6:53 PM	File	32 KB
COPYINGLESSER	9/29/2023 6:53 PM	File	8 KB
Keypad_MCP.cpp	9/29/2023 6:53 PM	C++ Source	6 KB
Keypad_MCP.h	9/29/2023 6:53 PM	C/C++ Header	3 KB
keywords.txt	9/29/2023 6:53 PM	Text Document	1 KB
library.properties	9/29/2023 6:53 PM	Properties Source ...	1 KB

Add Arduino library

- Method 2



Type: All
Topic: All

AlPlc_Opta by Arduino

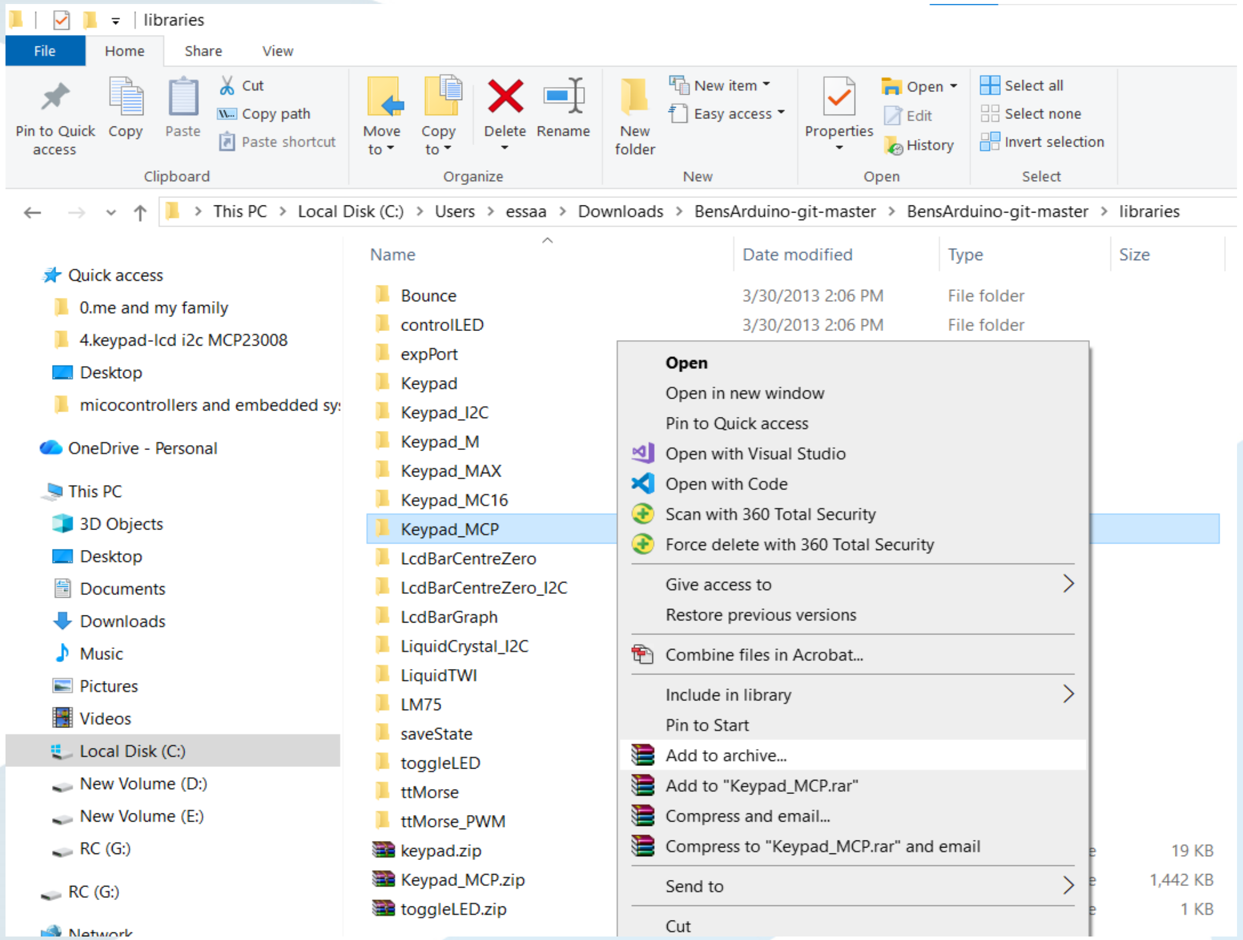
Arduino IDE PLC runtime library for Arduino Opta This is the runtime library and plugins fo...
More info

1.2.0 INSTALL

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly

}
```

Add Arduino library

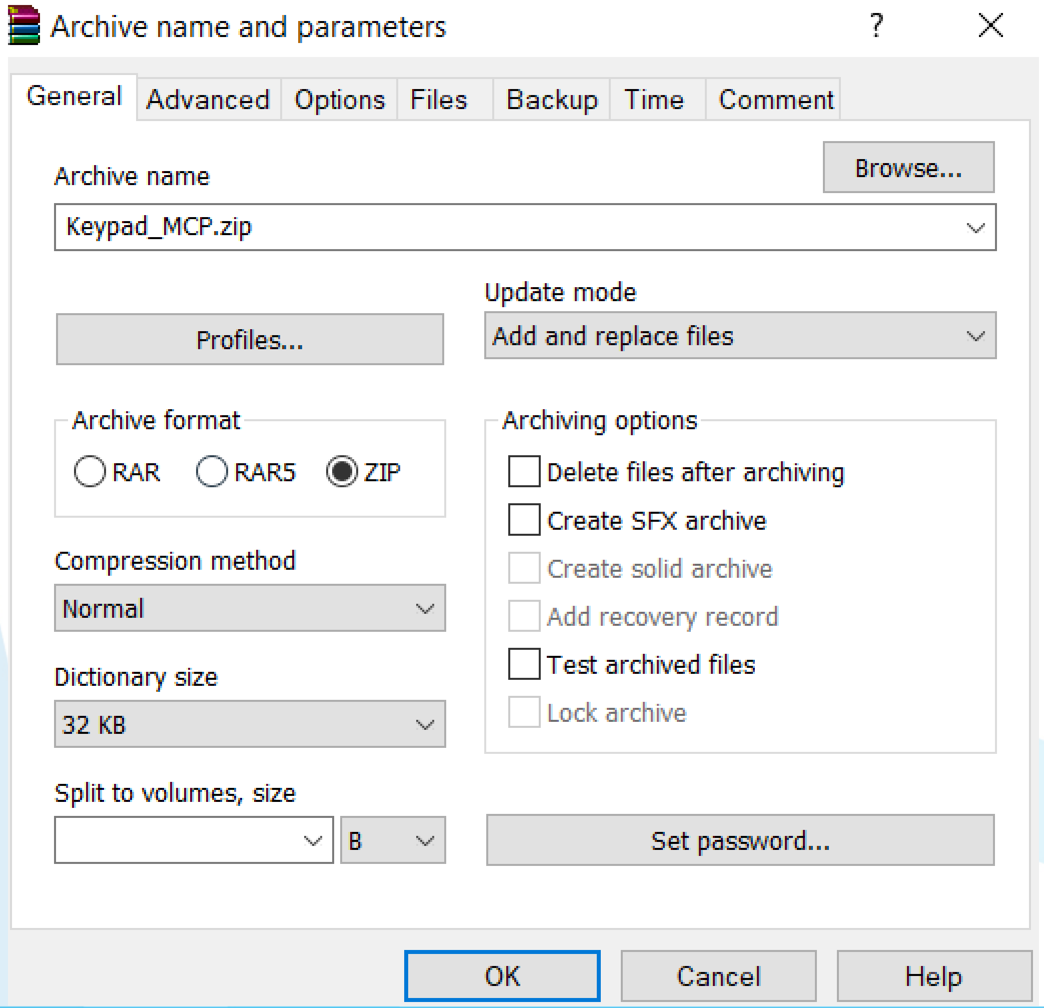
- Method 3



libraries
File Home Share View
Pin to Quick access Copy Paste Cut Copy path Paste shortcut
Clipboard
Move to Copy to Delete Rename
Organize
New folder New item Easy access
New
Properties Open Edit History
Open
Select all Select none Invert selection
Select
This PC > Local Disk (C:) > Users > essaa > Downloads > BensArduino-git-master > BensArduino-git-master > libraries
Quick access
0.me and my family
4.keypad-lcd i2c MCP23008
Desktop
micocontrollers and embedded sy
OneDrive - Personal
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
New Volume (D:)
New Volume (E:)
RC (G:)
RC (G:)
Network

Name	Date modified	Type	Size
Bounce	3/30/2013 2:06 PM	File folder	
controlLED	3/30/2013 2:06 PM	File folder	
expPort			
Keypad			
Keypad_I2C			
Keypad_M			
Keypad_MAX			
Keypad_MC16			
Keypad_MCP			
LcdBarCentreZero			
LcdBarCentreZero_I2C			
LcdBarGraph			
LiquidCrystal_I2C			
LiquidTWI			
LM75			
saveState			
toggleLED			
ttMorse			
ttMorse_PWM			
keypad.zip			19 KB
Keypad_MCP.zip			1,442 KB
toggleLED.zip			1 KB

Open
Open in new window
Pin to Quick access
Open with Visual Studio
Open with Code
Scan with 360 Total Security
Force delete with 360 Total Security
Give access to
Restore previous versions
Combine files in Acrobat...
Include in library
Pin to Start
Add to archive...
Add to "Keypad_MCP.rar"
Compress and email...
Compress to "Keypad_MCP.rar" and email
Send to
Cut



Archive name and parameters
General Advanced Options Files Backup Time Comment
Browse...
Profiles...
Set password...
OK Cancel Help

Archive name: Keypad_MCP.zip

Update mode: Add and replace files

Archive format: RAR RAR5 [✓ ZIP]

Compression method: Normal

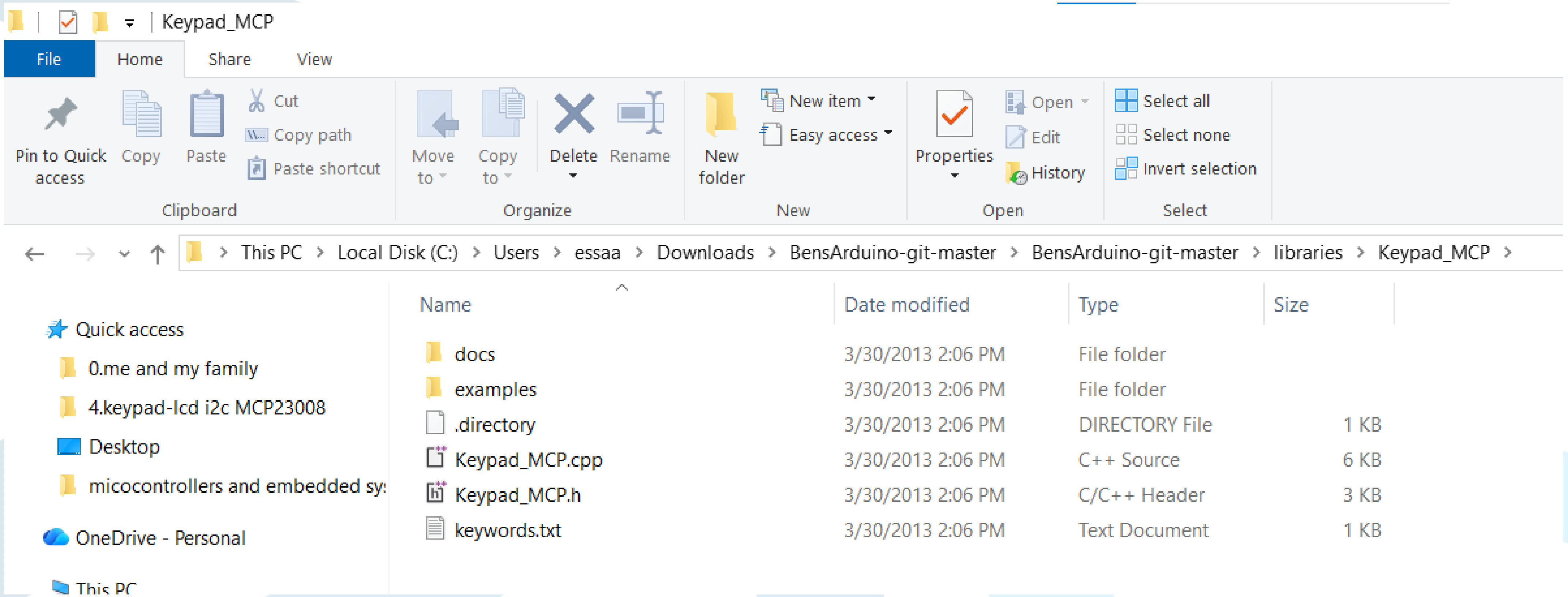
Dictionary size: 32 KB

Split to volumes, size: B

Archiving options:
- ☐ Delete files after archiving
- ☐ Create SFX archive
- ☐ Create solid archive
- ☐ Add recovery record
- ☐ Test archived files
- ☐ Lock archive

Add Arduino library

- Method 3

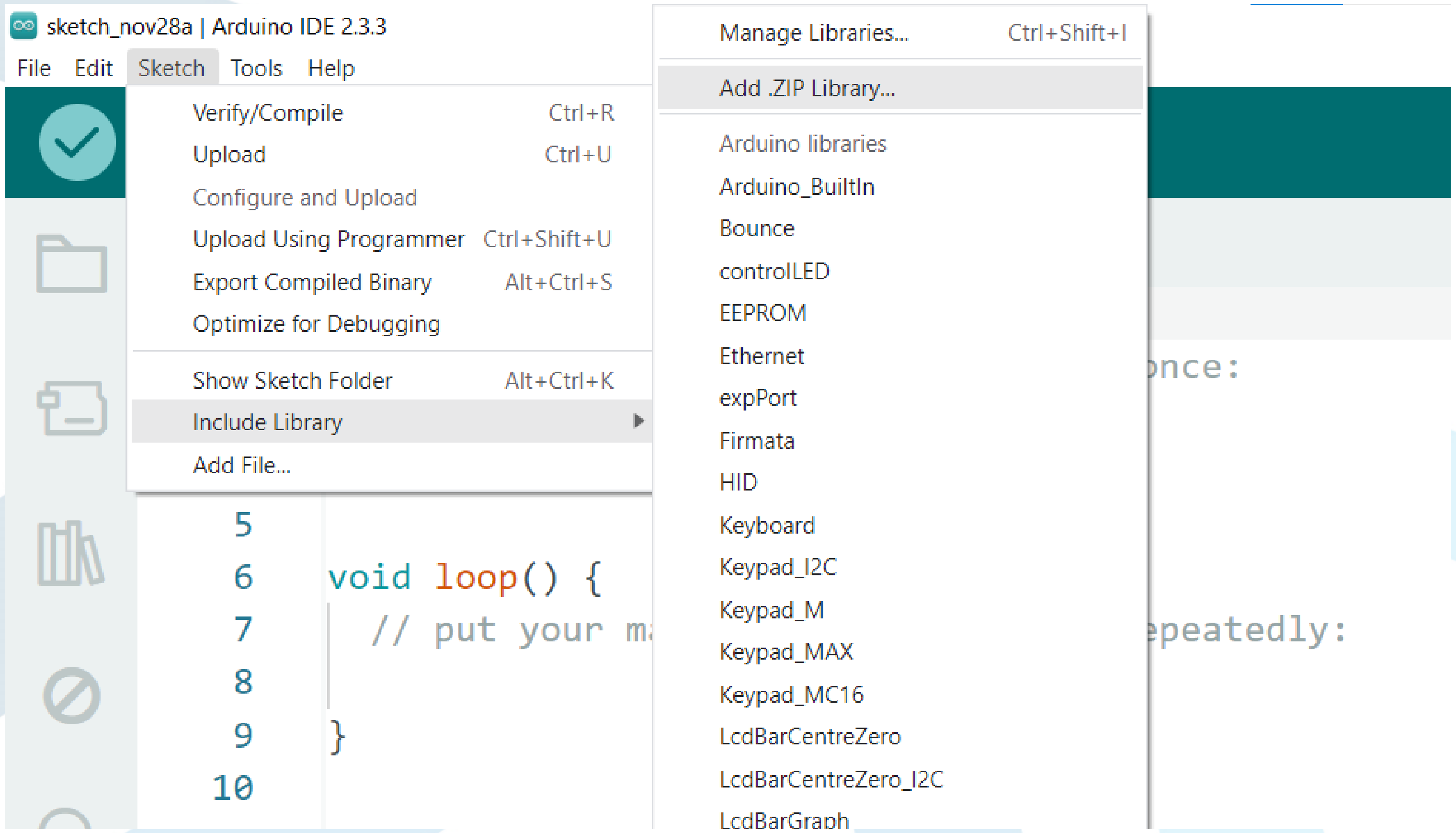


This PC > Local Disk (C:) > Users > essaa > Downloads > BensArduino-git-master > BensArduino-git-master > libraries > Keypad_MCP >

Name	Date modified	Type	Size
docs	3/30/2013 2:06 PM	File folder	
examples	3/30/2013 2:06 PM	File folder	
.directory	3/30/2013 2:06 PM	DIRECTORY File	1 KB
Keypad_MCP.cpp	3/30/2013 2:06 PM	C++ Source	6 KB
Keypad_MCP.h	3/30/2013 2:06 PM	C/C++ Header	3 KB
keywords.txt	3/30/2013 2:06 PM	Text Document	1 KB

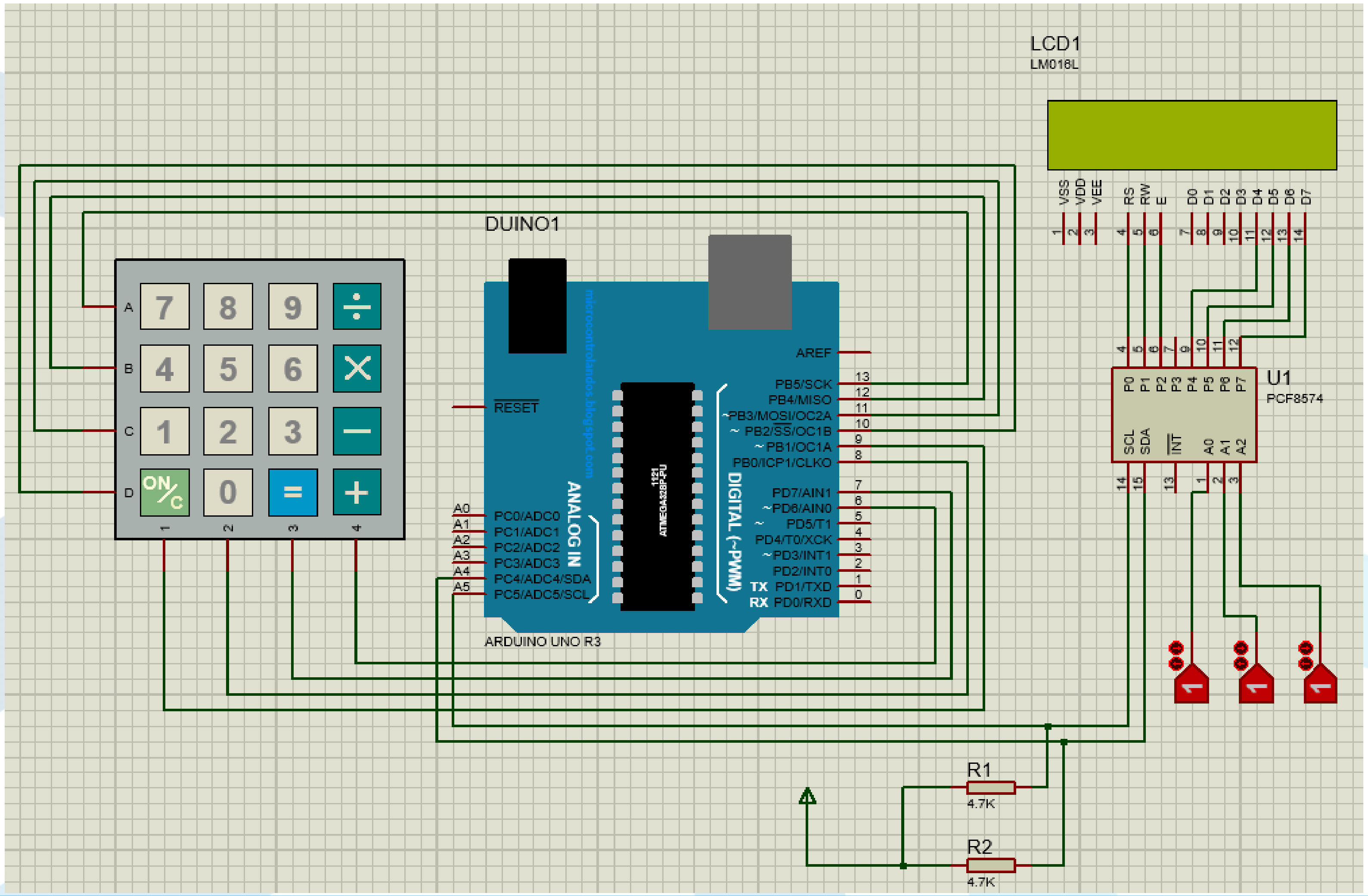
Add Arduino library

- Method 3



sketch_nov28a | Arduino IDE 2.3.3

```
void loop() {
  // put your m                                   epeatedly:

}
```


LCD1
LM016L
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7
DUINO1
ARDUINO UNO R3
AREF
RESET
PB5/SCK
PB4/MISO
~PB3/MOSI/OC2A
~ PB2/SS/OC1B
~ PB1/OC1A
PB0/ICP1/CLKO
PD7/AIN1
~PD6/AIN0
~ PD5/T1
PD4/T0/XCK
~PD3/INT1
PD2/INT0
TX PD1/TXD
RX PD0/RXD
DIGITAL (~PWM)
ANALOG IN
PC0/ADC0
PC1/ADC1
PC2/ADC2
PC3/ADC3
PC4/ADC4/SDA
PC5/ADC5/SCL
A0 A1 A2 A3 A4 A5
U1
PCF8574
P0 P1 P2 P3 P4 P5 P6 P7
SCL SDA INT A0 A1 A2
R1
4.7K
R2
4.7K
7 8 9 ÷
4 5 6 ×
1 2 3 −
ON/C 0 = +
A B C D

code

```
// Include Arduino Wire library for I2C
#include <Wire.h>
// Include LCD display library for I2C
#include <LiquidCrystal_I2C.h>
// Include Keypad library
#include <Keypad.h>

// Length of password + 1 for null character
#define Password_Length 8
// Character to hold password input
char Data[Password_Length];
// Password
char Master[Password_Length] = "0123456";
```

```
// Pin connected to lock relay input
int lockOutput = 0;

// Counter for character entries
byte data_count = 0;

// Character to hold key input
char customKey;

// Constants for row and column sizes
const byte ROWS = 4;
const byte COLS = 4;
```

- Master[0] = '0'
- Master[1] = '1'
- Master[2] = '2'
- Master[3] = '3'
- Master[4] = '4'
- Master[5] = '5'
- Master[6] = '6'
- Master[7] = '\0' (the null terminator)

```
// Array to represent keys on keypad
char hexaKeys[ROWS][COLS] = {


  {'7','8','9', '/'},
  {'4','5','6','x'},
  {'1','2','3','-'},
  {'*','0','#','+'}
};
```

```
// Connections to Arduino
byte rowPins[ROWS] = {13, 12, 11, 10};
byte colPins[COLS] = {9, 8, 7, 6};


// Create keypad object
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS,
COLS);


// Create LCD object
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

```
void setup() {
  // Setup LCD with backlight and initialize
  lcd.backlight();
  lcd.init();


  // Set lockOutput as an OUTPUT pin
  pinMode(lockOutput, OUTPUT);
}


void loop() {


  // Initialize LCD and print
  lcd.setCursor(0, 0);
  lcd.print("Enter Password:");


  // Look for keypress
  customKey = customKeypad.getKey();
```

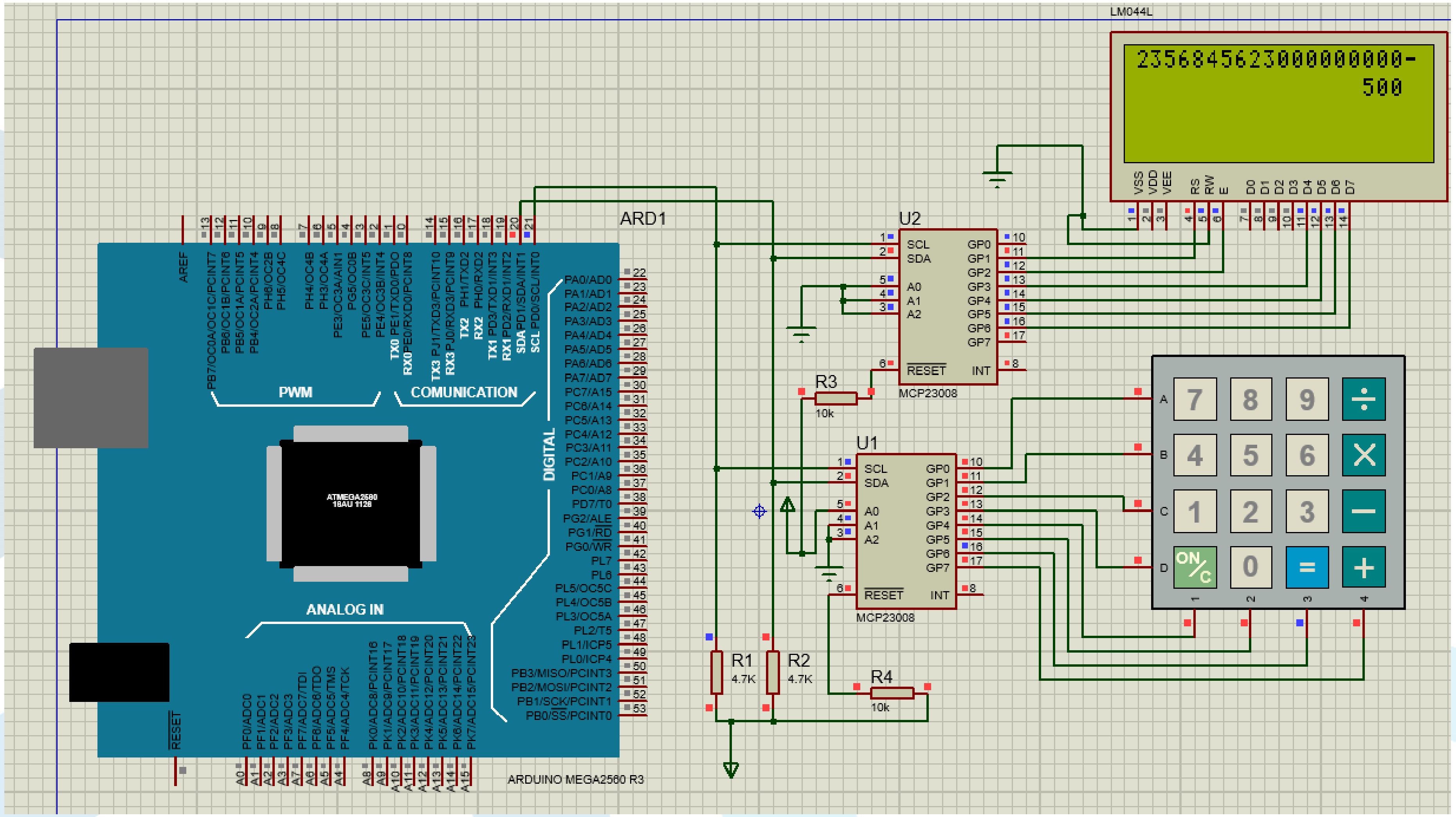
```
  if (customKey) {
    // Enter keypress into array and increment counte
    Data[data_count] = customKey;
    lcd.setCursor(data_count, 1);
    lcd.print(Data[data_count]);
    data_count++;
  }


  // See if we have reached the password length
  if (data_count == Password_Length - 1) {
    lcd.clear();


    if (!strcmp(Data, Master)) {
      // Password is correct
      lcd.print("Correct");
      // Turn on relay for 5 seconds
      digitalWrite(lockOutput, HIGH);
      delay(5000);
      digitalWrite(lockOutput, LOW);
    }
    else {
      // Password is incorrect
      lcd.print("Incorrect");
      delay(1000);
    }


    // Clear data and LCD display
    lcd.clear();
    clearData();
  }
}
```

```
void clearData() {
  // Go through array and clear data
  while (data_count != 0) {
    Data[data_count--] = 0;
  }
  return;
}
```

LM044L
20*4

MCP23008

Keypad small calculator



```
//#include <I2CKeyPad.h>
#include <Wire.h>
#include <Keypad_MCP.h>
//#include <Keypad.h>    // required; download from
http://playground.arduino.cc/uploads/Code/keypad.zip
#include <LiquidTWI2.h>


#define LCD_I2CADDR 0x20       // Connect via i2c,
default address 0x20
                               // Set A0, A1, A2 to

ground of 1st MCP23008
#define Keypad_I2CADDR 0x21    // connect pin A0 of 2nd
MCP23008 to 5V for address 0x21
                               // A1, A2 to ground

const byte ROWS = 4; // four rows
const byte COLS = 4; // four columns


//define the keymap
char keys [ROWS] [COLS] = {
  {'7', '8', '9', '/'},
  {'4', '5', '6', '*'},
  {'1', '2', '3', '-'},
  {'X', '0', '=', '+'}
};
```

```
byte rowPins[ROWS] = {0, 1, 2, 3}; //connect to the
row pinouts of the keypad
byte colPins[COLS] = {4, 5, 6, 7}; //connect to the
column pinouts of the keypad


LiquidTWI2 lcd(LCD_I2CADDR);
Keypad_MCP kpd( makeKeymap(keys), rowPins, colPins,
ROWS, COLS, Keypad_I2CADDR );


//variables declaration
boolean valOnePresent = false;
boolean next = false;
boolean final = false;
String num1, num2;
int ans;
char op;
void setup(){

  lcd.setMCPType(LTI_TYPE_MCP23008); // must be
called before begin()
  lcd.begin(20,4);                  // using 20 x 4
LCD display
  Wire.begin();
  kpd.begin( makeKeymap(keys) );
  lcd.home ();                    // go home
  lcd.setCursor(0,0);
  lcd.print(F("Using LiquidTWI2"));
  delay(2000);                    // 2 seconds warm-up
time
  lcd.clear();         // clears the LCD screen and
positions the cursor in the upper-left corner.
}
```

```
void loop(){
  char key = kpd.getKey();
//
if (key != NO_KEY &&
(key=='1'||key=='2'||key=='3'||key=='4'||key=='
5'||key=='6'||key=='7'||key=='8'||key=='9'||key
=='0')){
    if (valOnePresent != true){
      num1 = num1 + key;
      int numLength = num1.length();
      lcd.setCursor(19 - numLength,
0);         //to adjust one whitespace for
operator
      lcd.print(num1);
    }
    else {
      num2 = num2 + key;
      int numLength = num2.length();
      lcd.setCursor(19 - numLength, 1);
      lcd.print(num2);
      final = true;
    }
  }
```

```
  else if (valOnePresent == false && key !=
NO_KEY && (key == '/' || key == '*' || key == '-
' || key == '+')){
    if (valOnePresent == false){
      valOnePresent = true;
      op = key;
      lcd.setCursor(19,0);          //opera
tor on right corner
      lcd.print(op);
    }
  }


  else if (final == true && key != NO_KEY && key
== '='){
    if (op == '+'){
      ans = num1.toInt() + num2.toInt();
      //lcd.clear();
      lcd.setCursor(10,3);
      lcd.autoscroll();
      lcd.print(ans);
      lcd.noAutoscroll();
    }
else if (op == '-'){
      ans = num1.toInt() - num2.toInt();
      //lcd.clear();
      lcd.setCursor(10,3);
      lcd.autoscroll();
      lcd.print(ans);
      lcd.noAutoscroll();
    }
    else if (op == '*'){
      ans = num1.toInt() * num2.toInt();
      //lcd.clear();
      lcd.setCursor(10,3);
      lcd.autoscroll();
      lcd.print(ans);
      lcd.noAutoscroll();
    }
    else if (op == '/'){
      ans = num1.toInt() / num2.toInt();
      //lcd.clear();
      lcd.setCursor(10,3);
      lcd.autoscroll();
      lcd.print(ans);
      lcd.noAutoscroll();
    }
  }
```

```
else if (key != NO_KEY && key == 'X'){
    lcd.clear();
    valOnePresent = false;
    final = false;
    num1 = "";
    num2 = "";
    ans = 0;
    op = ' ';
  }
}
```

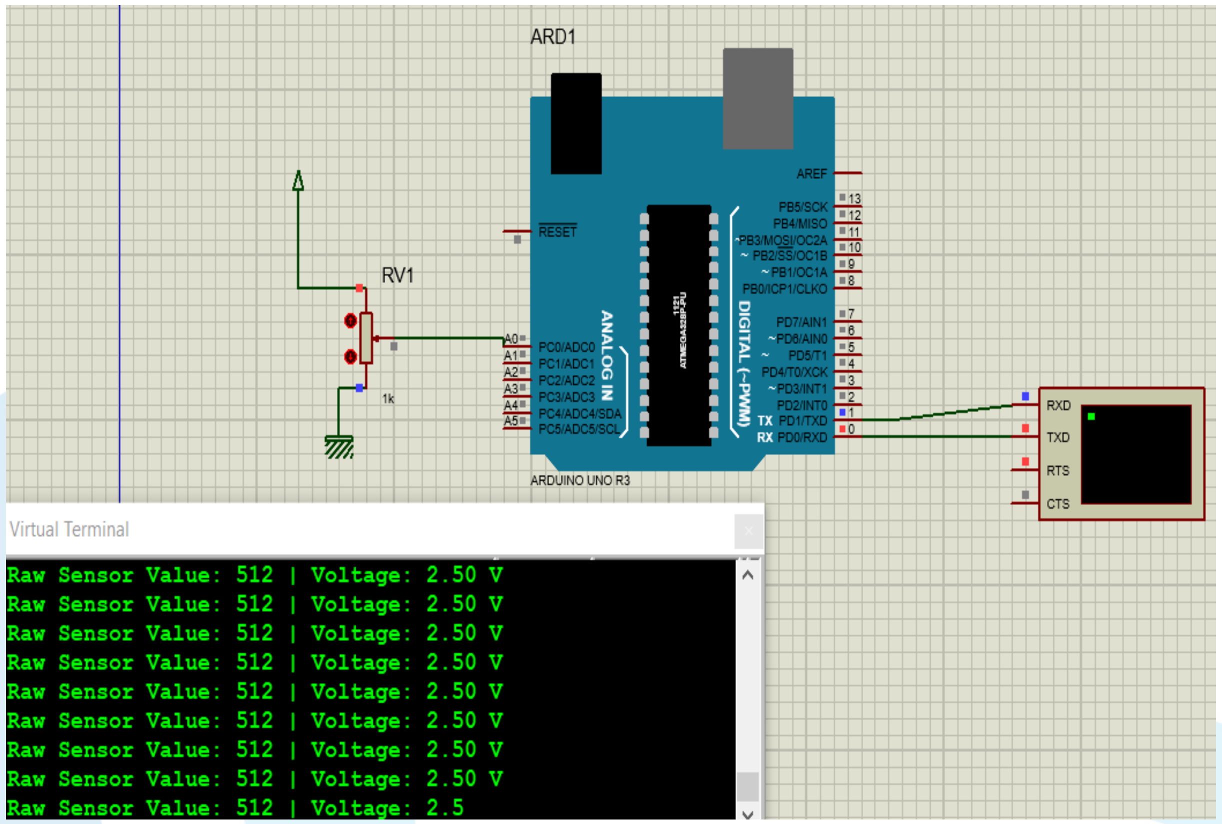
Analog Input

1. **Digital Pins:** Can read only two states: HIGH (approx. 5V) or LOW (approx. 0V).
2. **Analog Pins:** Can read a continuous range of voltage values (typically from 0V to 5V).
3. On the Arduino Mega, these pins have **10-bit resolution**. This means they convert the input voltage into a digital value ranging from **0 to 1023**.
4. **0** represents 0 Volts.
5. **1023** represents 5 Volts (or the AREF voltage if configured differently).

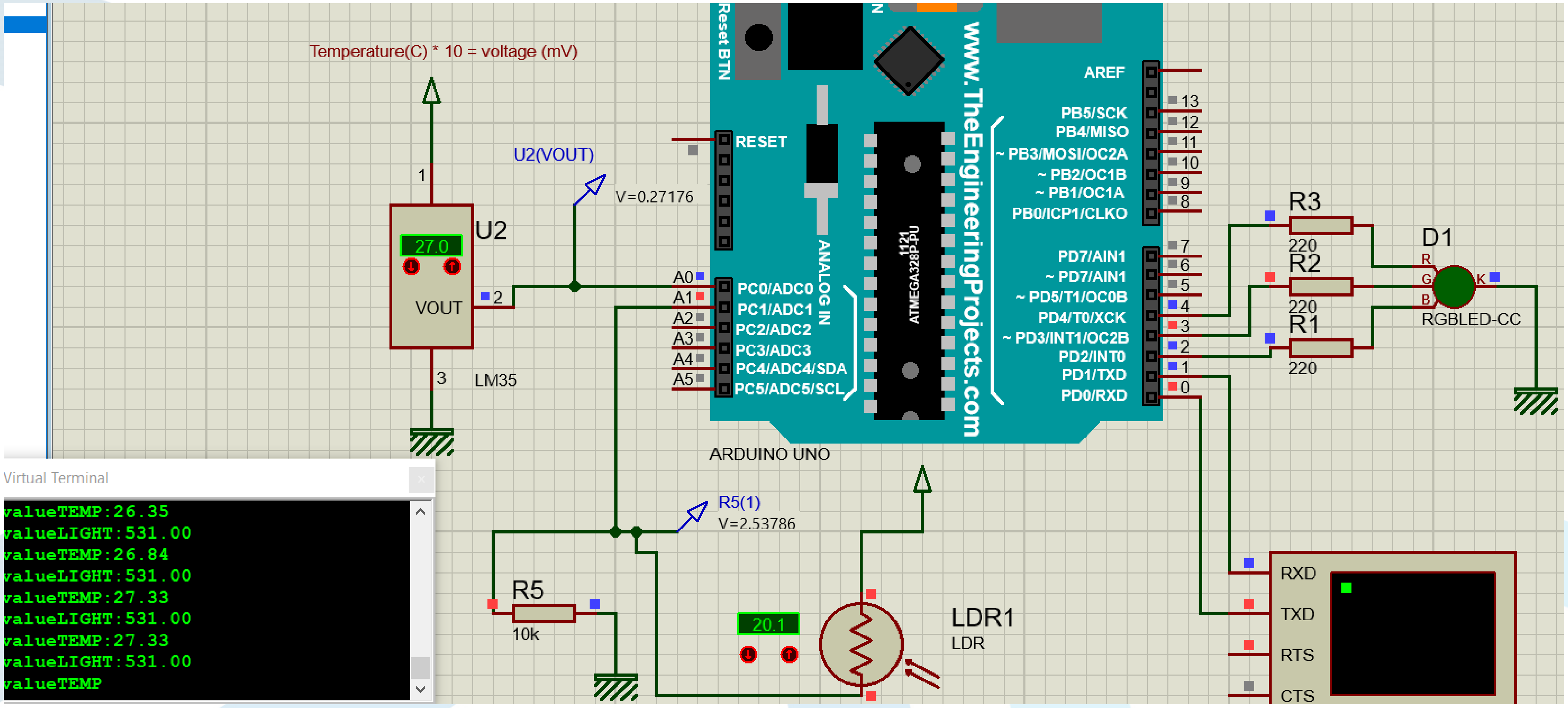
Required Components:

1. **Arduino uno.**
2. **10k Ohm Potentiometer (Pot)**

```
const int analogPin = A0;
void setup() {
  Serial.begin(9600);}
void loop() {
  // Read the value from the specified analog
pin (analogPin)
  // The analogRead() function returns a value
from 0 to 1023
  int sensorValue = analogRead(analogPin);
  Serial.print("Raw Sensor Value: ");
  Serial.print(sensorValue);
  //
===============================================
==========
  // Convert the raw value to voltage
  // Arduino reads 0-1023 for 0-5V.
  // So, each unit (1) in the reading
corresponds to (5.0V / 1024) Volts.
  //
===============================================
==========
  float voltage = sensorValue * (5.0 / 1024.0);
  Serial.print(" | Voltage: ");
  Serial.print(voltage);
  Serial.println(" V");
  delay(100);}
```

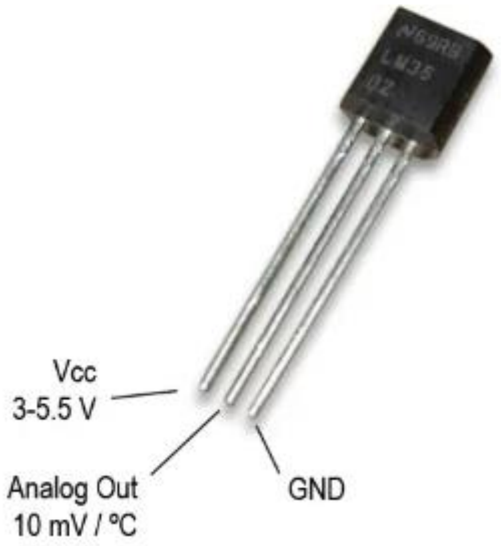


```
Raw Sensor Value: 512 | Voltage: 2.50 V
Raw Sensor Value: 512 | Voltage: 2.50 V
Raw Sensor Value: 512 | Voltage: 2.50 V
Raw Sensor Value: 512 | Voltage: 2.50 V
Raw Sensor Value: 512 | Voltage: 2.50 V
Raw Sensor Value: 512 | Voltage: 2.50 V
Raw Sensor Value: 512 | Voltage: 2.50 V
Raw Sensor Value: 512 | Voltage: 2.50 V
Raw Sensor Value: 512 | Voltage: 2.5
```



```
valueTEMP:26.35
valueLIGHT:531.00
valueTEMP:26.84
valueLIGHT:531.00
valueTEMP:27.33
valueLIGHT:531.00
valueTEMP:27.33
valueLIGHT:531.00
valueTEMP
```

حساس الحرارة LM35



The LM35 produces an analog voltage output.

•At 0°C, the output is 0V.

•At 1°C, the output is 10mV (0.01V).

•At 25°C, the output is 250mV (0.25V).

•At 100°C, the output is 1000mV (1V).

When you connect it to an Arduino's analog input, the Arduino reads this voltage and converts it into a digital value (0-1023). We then use a simple formula to convert that digital value back into temperature in Celsius and Fahrenheit.

Key Features of the LM35:

- Linear Celsius Scale Output: Its output voltage changes by 10 mV for every 1°C change in temperature.
- No External Calibration: It's factory-calibrated and doesn't require any external calibration.
- Direct Celsius Reading: It directly outputs a voltage proportional to Celsius, making calculations simple.
- Operating Range: Typically operates from -55°C to +150°C (depending on the specific package).
- Low Power Consumption: Very efficient.
- Simple to Use: Only requires 3 pins.

LDR (Light Dependent Resistor)
NON-LINEAR sensor

The LDR itself is a variable resistor whose resistance changes with the intensity of light falling on it.

To read this change with an analog pin, we need to convert the resistance change into a voltage change.

We do this using a **voltage divider circuit**.

LDR Behavior:

•In bright light, its resistance is low (e.g., a few hundred ohms).

•In darkness, its resistance is very high (e.g., several megaohms).

•**Voltage Divider:** We place the LDR in series with a fixed resistor. The voltage at the point between these two resistors will change as the LDR's resistance changes, and this is the voltage the Arduino reads.

```
// - High LDR value (closer to 1023) means MORE light.
// - Low LDR value (closer to 0) means LESS light
(darker).
```

• For example, if you double the light intensity, the resistance will *not* halve.

• Often, the relationship is inverse exponential or follows a power law.

Sensitivity: An LDR might be very sensitive to small changes in light when it's very dark (meaning a small increase in light causes a large drop in resistance).
However, in brighter conditions, it might require a much larger change in light to produce the same change in resistance.

NON-LINEAR sensor vs LINEAR sensor

A linear sensor (like the LM35 temperature sensor we just discussed) provides an output that is *directly proportional* to the measured physical quantity.
For the LM35, if the temperature doubles, its output voltage doubles (or changes by a consistent, constant factor per degree). This makes it very easy to convert the sensor's output directly into a meaningful unit.

Because LDRs are non-linear, if you need to measure light intensity in an absolute, precise unit like "lux," you often need to:

1. Calibrate the LDR across its operating range using a known lux meter.
2. Apply a more complex mathematical function (lookup table, polynomial, power function) in your code to linearize the readings.
3. Use a dedicated digital lux sensor (like the BH1750 or TSL2561) which provides a direct, linear reading in lux and often handles ambient light filtering internally.

For simple light/dark detection or relative light level changes, the non-linearity of an LDR is usually not a problem. But for precise measurements, it's a significant factor to consider.

```
const int BLED=2; //Blue LED on pin 9
const int GLED=3; //Green LED on pin 10
const int RLED=4; //Red LED on pin 11
const int TEMP=0; //Temp Sensor is on pin A0
const int LOWER_BOUND=25; //Lower Threshold
TEMP
const int UPPER_BOUND=30; //Upper Threshold
TEMP


const int LIGHT=1;          //light Sensor is
on pin A1
const int MIN_LIGHT=0; //Minimum expected light
value
const int MAX_LIGHT=1000; //Maximum Expected
Light value
double valueTEMP= 0, valueLIGHT=0; //Variable
to hold analog reading


void setup()
{
Serial.begin(9600);
pinMode(BLED, OUTPUT); //Set Blue LED as Output
pinMode(GLED, OUTPUT); //Set Green LED as
Output
pinMode(RLED, OUTPUT); //Set Red LED as Output
}
```

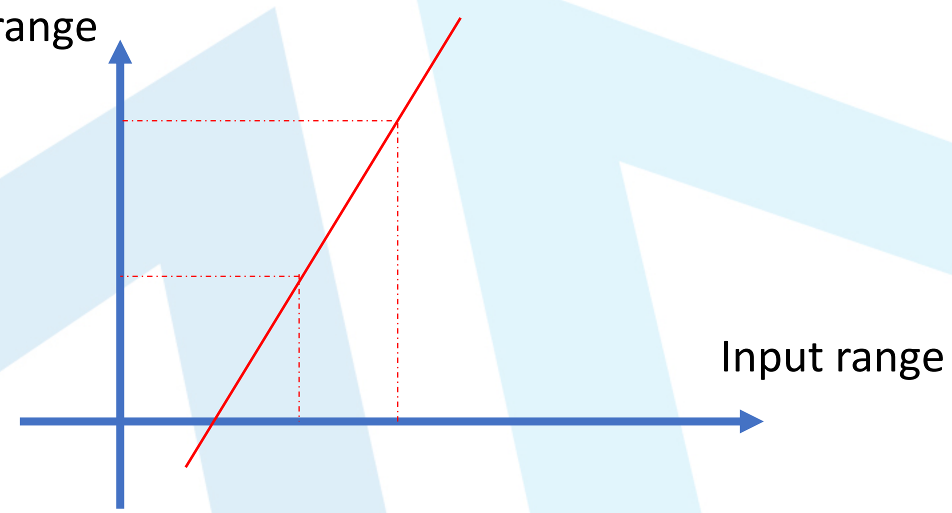
```
void loop()
{
valueTEMP= 0.488*analogRead(TEMP);
Serial.print("valueTEMP:");
Serial.println(valueTEMP);
if (valueTEMP< LOWER_BOUND)
{
digitalWrite(RLED, LOW);
digitalWrite(GLED, LOW);
digitalWrite(BLED, HIGH);
}
else if (valueTEMP> UPPER_BOUND)
{
digitalWrite(RLED, HIGH);
digitalWrite(GLED, LOW);
digitalWrite(BLED, LOW);
}
else
{
digitalWrite(RLED, LOW);
digitalWrite(GLED, HIGH);
digitalWrite(BLED, LOW);
}
valueLIGHT =analogRead(LIGHT);
valueLIGHT= map(valueLIGHT, MIN_LIGHT,
MAX_LIGHT, 0, 1023); //Map the light
reading
valueLIGHT= constrain(valueLIGHT, 0,
1023); //Constrain light value //
constrain a number between an upper
bound and a lower bound.
Serial.print("valueLIGHT:");
Serial.println(valueLIGHT);
delay(200);
}
```

the `map()` function in Arduino performs a linear interpolation.

This means it re-maps a number from one range to another by essentially drawing a straight line between the corresponding minimum and maximum values of the input and output ranges. The output value will be proportionally adjusted along this line based on the input value's position within its range.

```
int val = 20;
val = map(val, 0, 10, 0, 100);
```

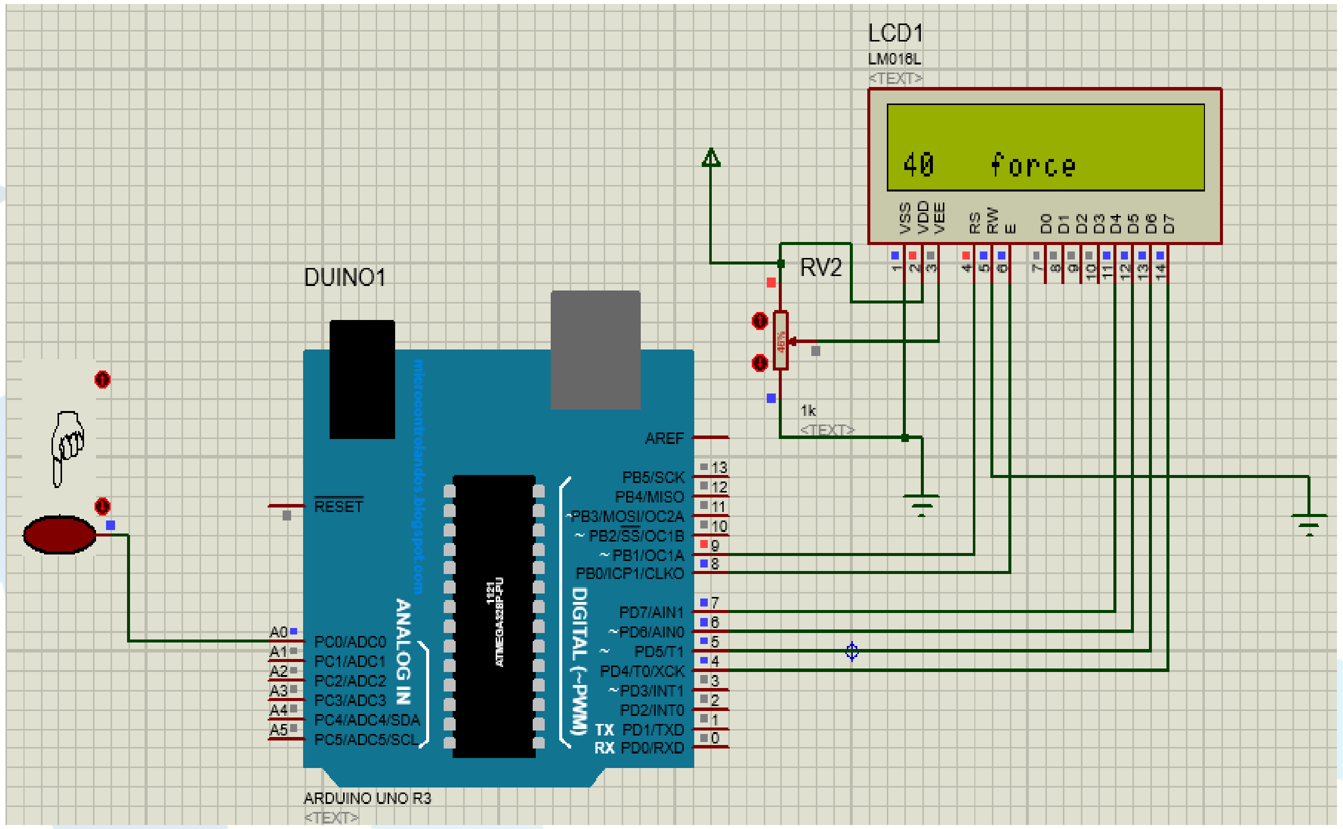
Although you set the upper bound of the value's range to 10, you passed an higher value than that and the function will linearly map it accordingly, resulting in an output of 200.



The `constrain()` function in Arduino

1. If the input value (`x`) is within the defined range (between `a` and `b`), the function returns `x` itself. This part is "linear" in the sense that the output directly matches the input.
2. If the input value (`x`) is less than the lower bound (`a`), the function returns `a`. This is a constant output, not a linear relationship.
3. If the input value (`x`) is greater than the upper bound (`b`), the function returns `b`. This is also a constant output.

```
#include <LiquidCrystal.h>
#define rs 9
#define en 8
#define d4 7
#define d5 6
#define d6 5
#define d7 4
#define sensorpin A0 // analog pin 0
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
//LiquidCrystal lcd(9, 8, 7, 6, 5, 4);

void setup() {
lcd.begin(16, 2);

lcd.setCursor(3,0);// 0 is first line
lcd.print("FORCE");

lcd.setCursor(0,1);// 1 is second line
lcd.print("DETECTOR");
delay(1000);
lcd.clear();
}
void loop() {
  int force=analogRead(sensorpin);
  int force_per= map(force, 0, 255, 0, 100);
  lcd.setCursor(5,1); // 1 is second line
  lcd.print("force");
  lcd.setCursor(0, 1);// 1 is second line
  lcd.print(force_per);
  delay(200);
}
```



LCD1
LM016L
40 force
RV2
1k
DUINO1
ARDUINO UNO R3
AREF
RESET
ANALOG IN
DIGITAL (~PWM)
A0
A1
A2
A3
A4
A5