I²C

Embedded Workshop 10/28/15 Rusty Cain

Set up for Workshop:

Please Sign in on Sheet. Please include your email.

While you are waiting for the Workshop to begin…

1. Make sure you are connected to the local Wifi
   Guest Password: **Welcome2DMS**
2. Make sure you have **Arduino IDE** installed and working:
   Download Arduino Version 1.6.4 or newer.
   Add the Wire.h library
   Sketch - Include Libraries - Manage Libraries - Wire?
3. **Copy Programs off the USB Memory stick.**
   Project1, Project2, Project3, Project4 & Extra Project

Parts Needed: (Ask about Parts kits)

Arduino Uno,
Breadboard & Wires,
3 - LED's,
3 - 330 ohm resistors,
1- TC74 Temp sensor,
1- I2C eeprom

Please sign in

Name	Member Y/N	Email address

I²C -TWI Bus

I²C Bus

The I²C bus was designed by Philips in the early ’80s to allow easy communication between components which reside on the **same** circuit board.

Philips Semiconductors migrated to NXP in 2006.

The Inter-Integrated Circuit name translates:
IIC, I2C I²C or TWI

TWI Bus

TWI stands for Two Wire Interface. This bus is identical to I²C. The name TWI was introduced by Atmel and other companies to avoid conflicts with trademark issues related to I²C.

Most significant features include:

- Only two bus lines are required
- No strict baud rate requirements like for instance with RS232.
- The master generates a bus clock.
    a. Bus can have different clock speeds.
        i. *CLOCK_SPEED_10_KHZ*
        ii. *CLOCK_SPEED_50_KHz*
        iii. *CLOCK_SPEED_100_KHZ*
        iv. *CLOCK_SPEED_400_KHZ*
- Simple master/slave relationships exist between all components
- Each device connected to the bus is software-addressable by a unique address
- I2C is a true multi-master bus providing arbitration and collision detection

I²C Overview

The devices on the I2C bus are either masters or slaves.

Master This is the device that generates clock, starts communication, sends I2C commands and stops communication

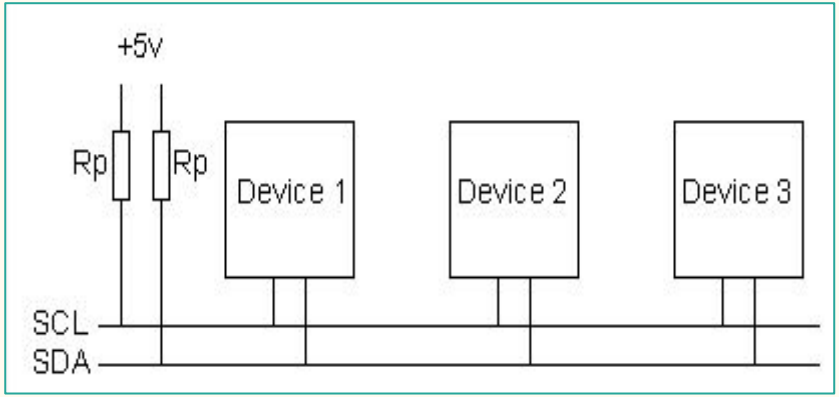
Slave This is the device that listens to the bus and is addressed by the master

Multi-master I2C can have more than one master and each can send commands

Arduino already has pull-up resistors on the SDA and SCL pins. All though this would not hurt the circuit it's not needed because when the Wire.h library is initialized it knows pins 4 and 5 are going to be used for I²C so it also activates the built-in pull-up resistors.

Bus Example: Refer to **Data sheet** for Resistor values

The I²C bus' master uses a device's 7-bit address to signal the component it wants to talk to and is shifted left to create an 8-bit value

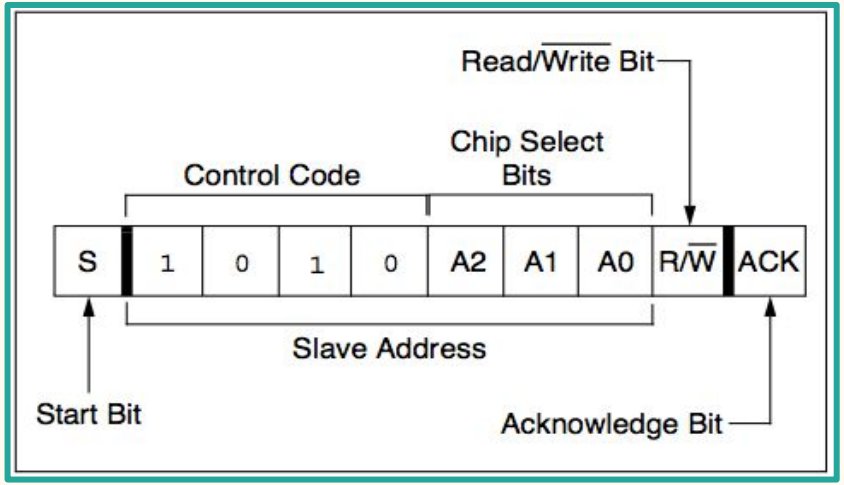


I²C Bus & Addressing

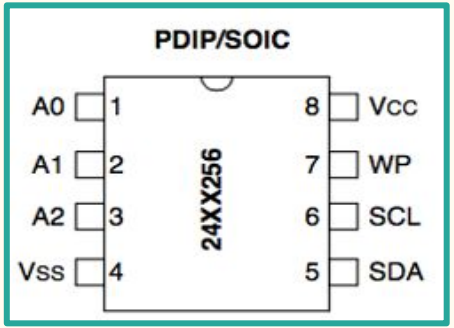
I²C bus uses a 7-bit address that is passed along with a read/write bit. Since a byte comprises eight bits.
The extra bit is used to indicate whether the signal is being sent by the master to the slave as a ‘write’ or a ‘read’.
Because of the I²C addresses being 7-bit numbers enables the bus to comprise up to 127 devices.
The 7-bit address is placed in bits one through seven.

The Arduino takes care of the last R/W bit for us depending on what function we’re using so as long as you’re using the standard Arduino Wire library.

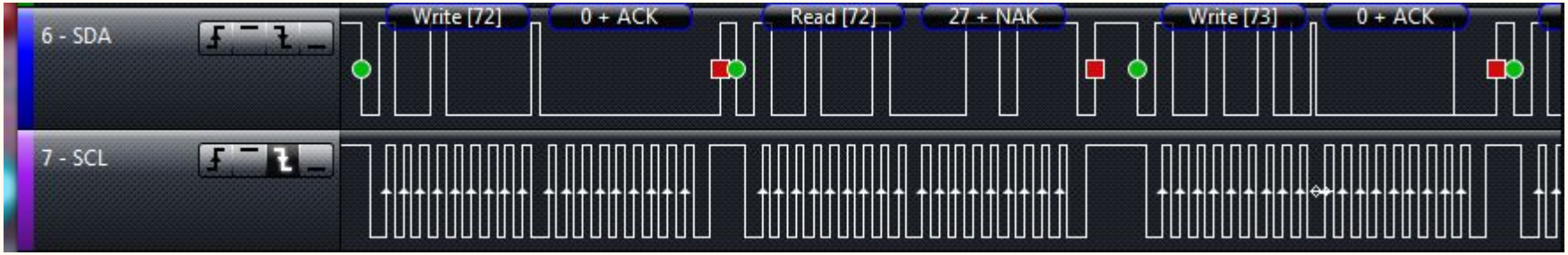
The Seven middle address bits.
The first four bits(Control Code) are hard-wired and can’t be changed.
The next three bits(A2,A1,A0) are the important bits and we can change them.
What address the chip will have is dependent on what we set these pins to.
Tie pins 1,2 and 3 on the 24LC256 to GND then the chip would have address 0×50 and if tie them all Vcc then the chip would have address 0×57 .




```
A0 = 1001 000
A1 = 1001 001
A2 = 1001 010
A3 = 1001 011
A4 = 1001 100
A5 = 1001 101
A6 = 1001 110
A7 = 1001 111
```



I²C Bus Data Transfer



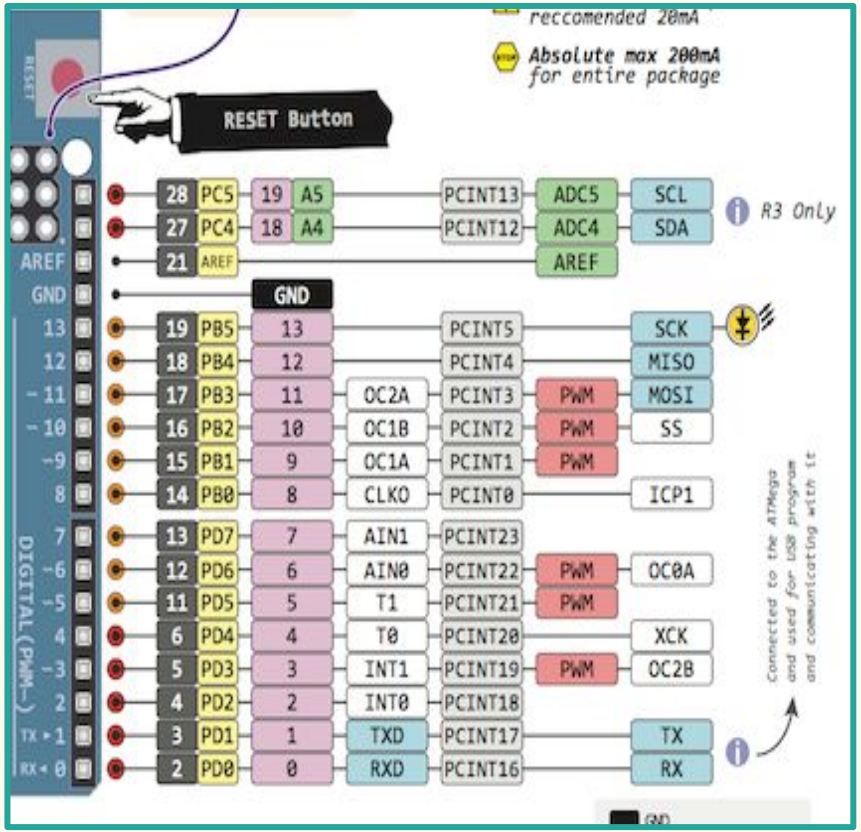
Data transfer sequence
1. Send the START bit (S).
2. Send the slave address (ADDR).
3. Send the Read(R)-1 / Write(W)-0 bit.
4. Wait for/Send an acknowledge bit (A).
5. Send/Receive the data byte (8 bits) (DATA).
6. Expect/Send acknowledge bit (A).
7. Send the STOP bit (P).

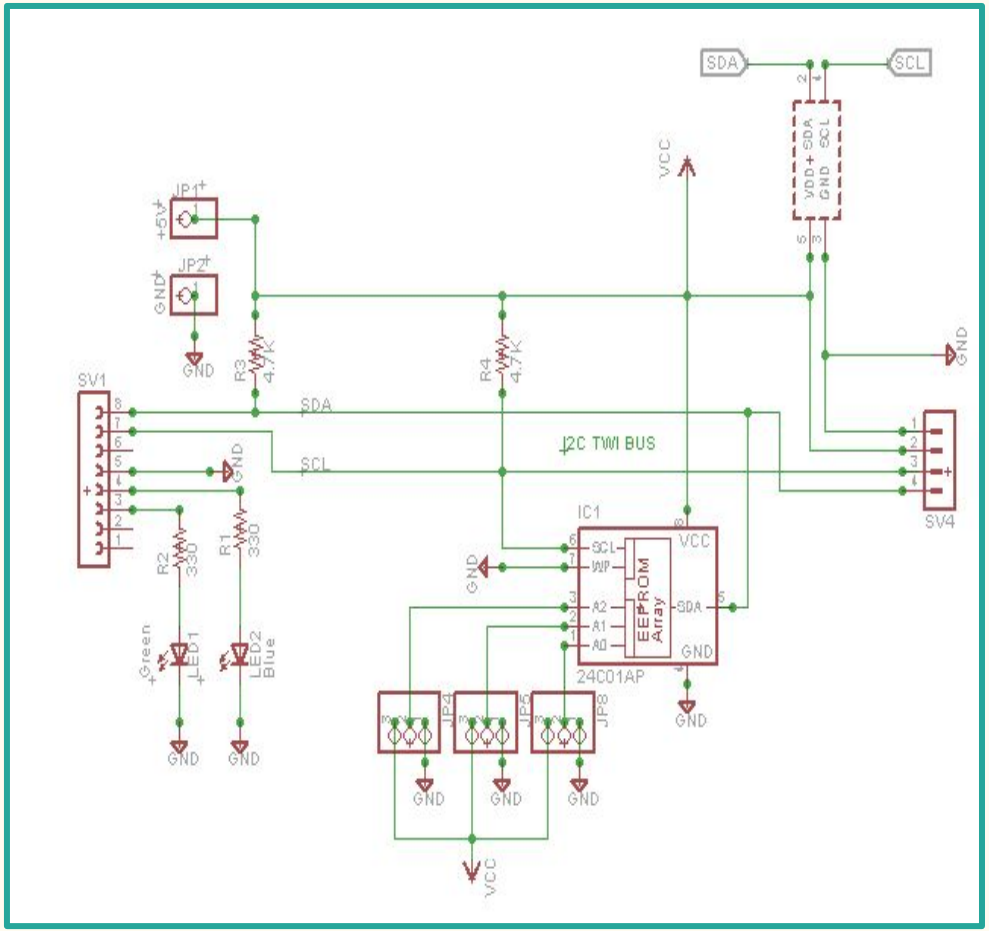
Repeated START (Sr)
The main reason that the Sr bit exists is in a multi master configuration where the current bus master does not want to release its mastership. Using the repeated start keeps the bus busy so that no other master can grab the bus.

Data Transfer from master to slave
A master device sends the sequence S ADDR W and then waits for an acknowledge bit (A) from the slave which the slave will only generate if its internal address matches the value sent by the master. If this happens then the master sends DATA and waits for acknowledge (A) from the slave. The master completes the byte transfer by generating a stop bit (P) (or repeated start).

Data transfer from slave to master
Instead of W, R is sent. After the data is transmitted from the slave to the master the **master** sends the acknowledge (A). If instead the master does not want any more data it must send a not-acknowledge which indicates to the slave that it should release the bus.

I²C PinOut Overview





I²C & Arduino

In order to use the I²C interface we need to include the Arduino standard Wire library `#include <Wire.h>`

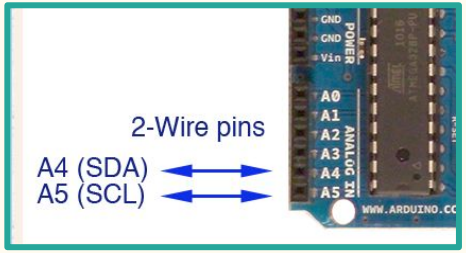
Need to define the address for your device

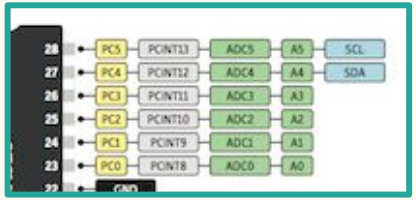
```
#define disk1 0x50  //Address of 24LC256
#define Temp 0x50   //Address of TC74A0
int address = 72;   //Decimal Address of TC74A2
```

Note: Arduino versions before 1.0 use **Wire.send** and **Wire.receive**

If you are using Arduino 1.0 and above then you need to use **Wire.write** and **Wire.read**

Analog port 4 (A4) = SDA (serial data)
Analog port 5 (A5) = SCL (serial clock)






```
#include <Wire.h>
#define disk1 0x50    //Address of 24LC256 eeprom chip

void setup(void)
{
  Serial.begin(9600);
  Wire.begin();
   unsigned int address = 0;
   writeEEPROM(disk1, address, 123);
  Serial.print(readEEPROM(disk1, address), DEC);
}
void loop(){}
void writeEEPROM(int deviceaddress, unsigned int eeaddress, byte
{
  Wire.beginTransmission(deviceaddress);
  Wire.write((int)(eeaddress >> 8));   // MSB
  Wire.write((int)(eeaddress & 0xFF)); // LSB
  Wire.write(data);
  Wire.endTransmission();
   delay(5);
}
```

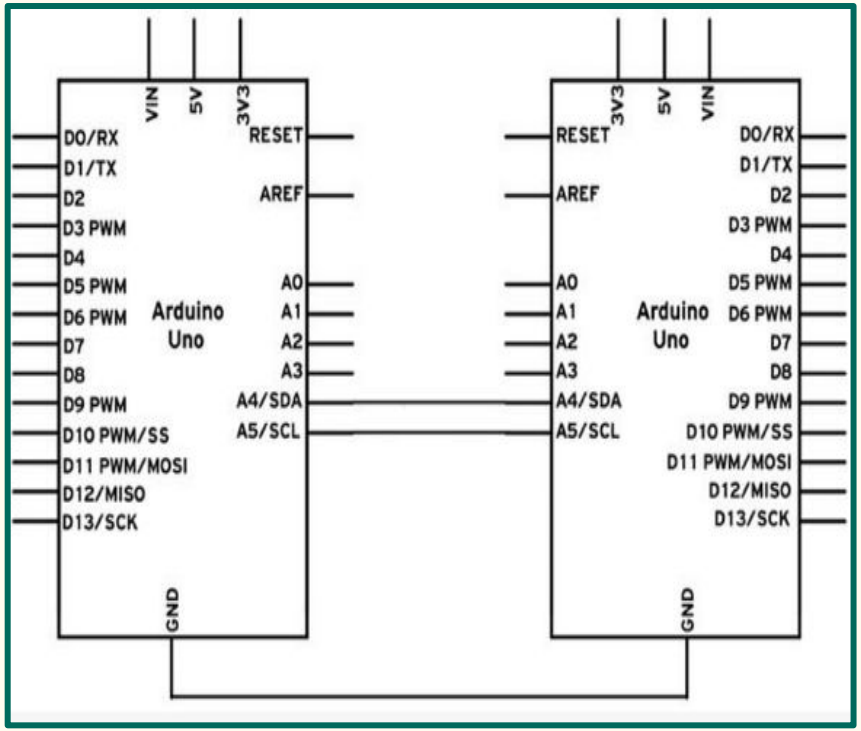
Communication Between Arduinos Using I²C

Arduino Board I2C pins

Uno, Pro Mini A4 (SDA), A5 (SCL)

Mega, Due 20 (SDA), 21 (SCL)

Leonardo, Yun 2 (SDA), 3 (SCL)



I²C Projects 1, 2, 3, 4, 5

Project 1

I²C Scanner Tool.

Project 2

Read TC74 Temperature Sensor & Display to console.

Project 3

Write to & Read from eeprom

Project 4

Read TC74 Temp and Write to eeprom.

Project 5

Use &View with Logic Analyzer.

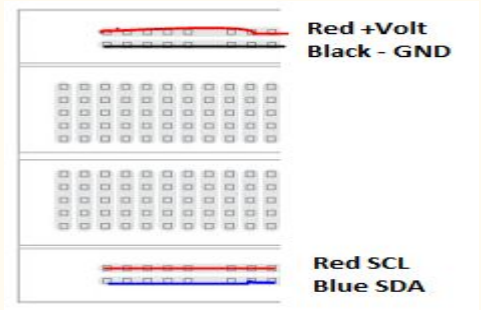
Project Extra

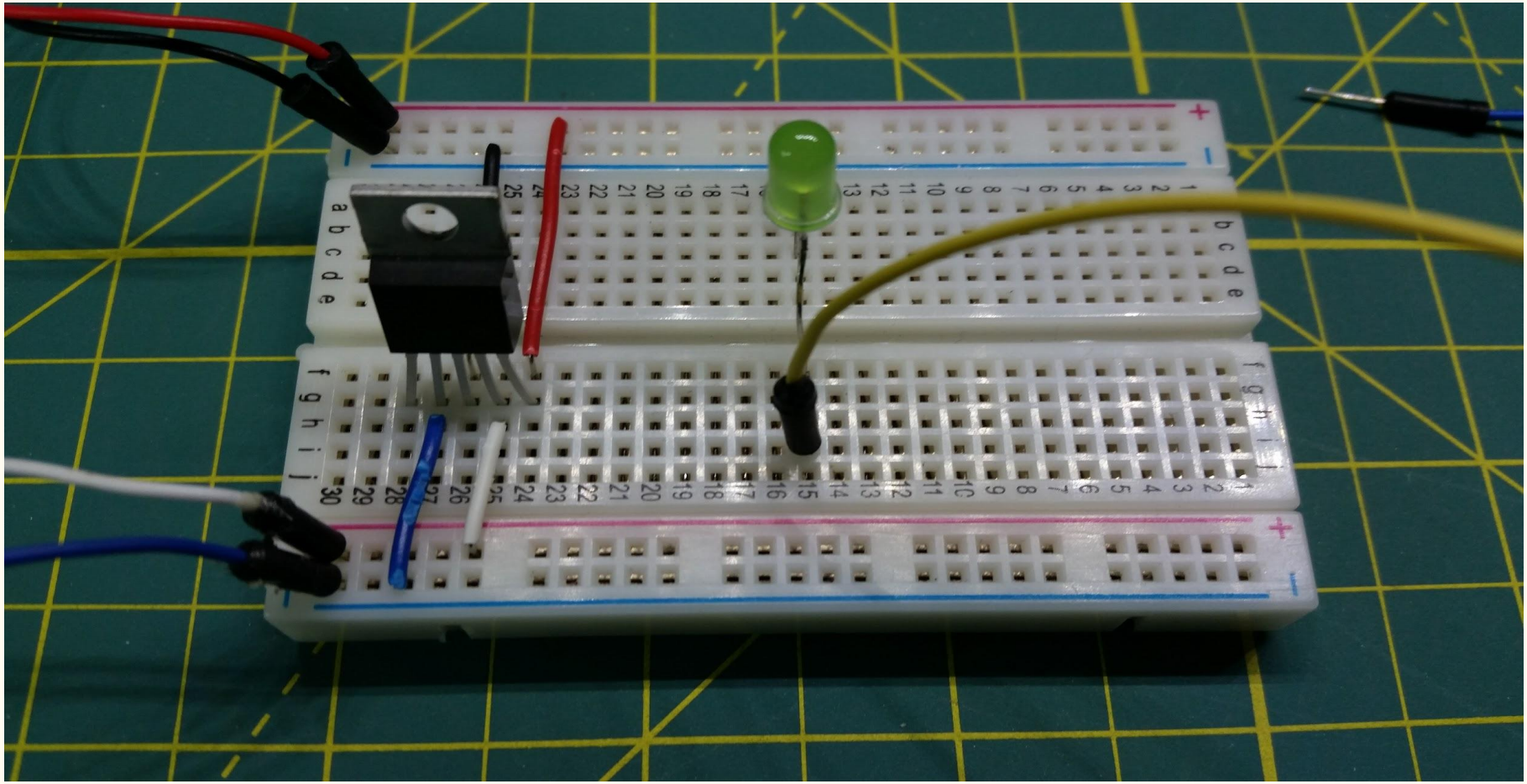
Connect 2 Arduino Uno together using I²C

Set up your breadboard with your I²C devices
Connect your breadboard to the Arduino.

TC74 Pinout

Arduino analog pin 4 to TC74 pin 2
Arduino analog pin 5 to TC74 pin 4
Arduino 5V to TC74 pin 5
Arduino GND to TC74 pin 3
Arduino Pin 12 to Green LED
LED Resistor to GND



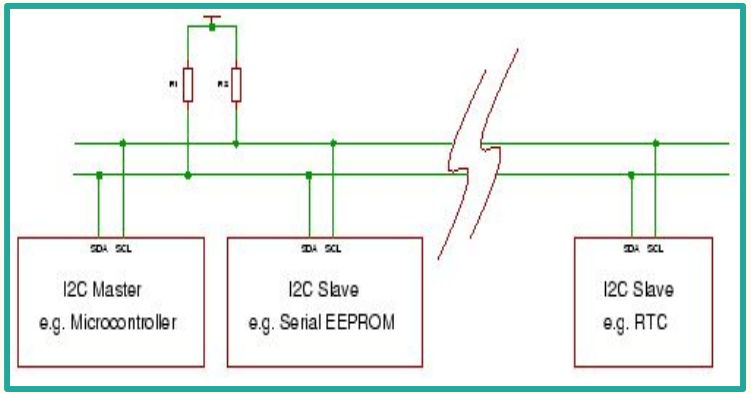


I2C Project 1 & 2 Wiring using TC74 Temperature Sensor

Project 1- I²C Scanner

Download I²C Scanner
http://playground.arduino.cc/Main/I2cScanner
Connect your I²C devices to your Arduino Bd.

This program will Scan for all devices on the I²C bus and report back all the device address found.




```
// I2C Scanner
#include <Wire.h>
void setup() {
  Serial.begin (115200);
  Serial.println ();
  Serial.println ("I2C scanner. Scanning ...");
  byte count = 0;
  Wire.begin();
  for (byte i = 8; i < 120; i++)
  {
    Wire.beginTransmission (i);
    if (Wire.endTransmission () == 0)
      {
      Serial.print ("Found address: ");
      Serial.print (i, DEC);
      Serial.print (" (0x");
      Serial.print (i, HEX);
      Serial.println (")");
      count++;
      delay (1);  // maybe unneeded?
      } // end of good response
  } // end of for loop
  Serial.println ("Done.");
  Serial.print ("Found ");
  Serial.print (count, DEC);
  Serial.println (" device(s).");
}  // end of setup
void loop() {}
```

Project 2 Read TC74 Temperature Sensor

TC74Ax

TC74 always operates as a Slave
TC74A0 in TO-220 package.
A0 corresponds to the device address 1001 000 (72)
A1 corresponds to the device address 1001 001 (73)

Pinout

Arduino analog pin 4 to TC74 pin 2
Arduino analog pin 5 to TC74 pin 4
Arduino 5V to TC74 pin 5
Arduino GND to TC74 pin 3

TC74A0-5.0VCT	1001 000
TC74A1-5.0VCT	1001 001
TC74A2-5.0VCT	1001 010
TC74A3-5.0VCT	1001 011
TC74A4-5.0VCT	1001 100
TC74A5-5.0VCT	1001 101*
TC74A6-5.0VCT	1001 110
TC74A7-5.0VCT	1001 111

Refer to the TC74 datasheet

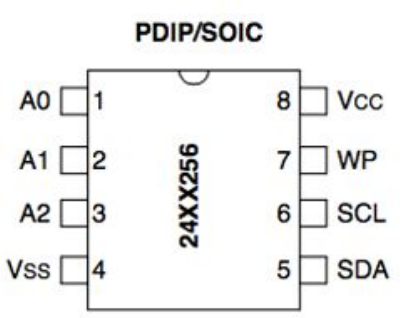
```
Include Wire I2C library
#include <Wire.h>
int temp_address = 73; 1001 001 written as decimal number
void setup()
{
Start serial communication at 9600 baud
 Serial.begin(9600);
Create a Wire object
 Wire.begin();
}
void loop()
{
Send a request & start talking to the device at address 73
 Wire.beginTransmission(temp_address);
Send a bit asking for register zero, the data register
 Wire.write(0);
 Complete Transmission
 Wire.endTransmission();
Read the temperature from the device using the read bit
Request 1 Byte from the specified address
 Wire.requestFrom(temp_address, 1);
 Wait for response
 while(Wire.available() == 0);
 Get the temp and read it into a variable
 int c = Wire.read();
 Math to convert the Celsius to Fahrenheit
 int f = round(c*9.0/5.0 +32.0);
 Temperature in degrees C and F to the serial monitor display
 Serial.print("Testing Temp  ");
 Serial.print(c);
 Serial.print("C ");
 Serial.print(f);
 Serial.println("F");
  delay(500);
}
```

Project 3 -Write & Read to eeprom

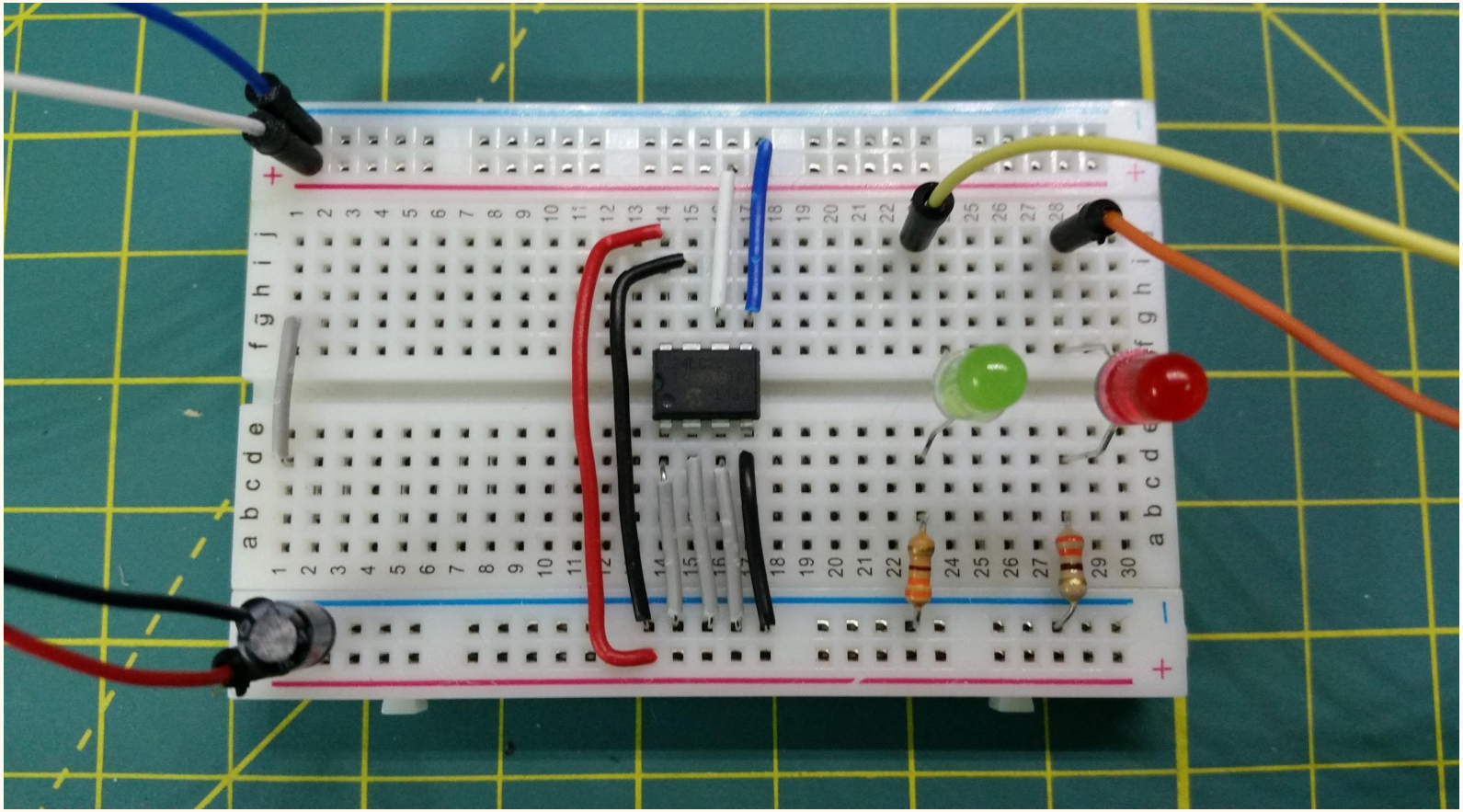
Connect 24LC256 eeprom To Arduino Uno

Arduino analog pin 11 to LED on Bread Bd.
Arduino analog pin 12 to LED on Bread Bd.
Arduino analog pin 4 (A4) to EEPROM pin 5. SDA
Arduino analog pin 5 (A5) to EEPROM pin 6. SCL
Arduino 5V to EEPROM pin 8 VCC.
Arduino GND to EEPROM pin 1,2,3,4
Pin 7 of the EEPROM tie it to GND otherwise the EEPROM will be write protected

24LC256 Data Sheet




```
#include <Wire.h>
#define disk1 0x50    //Address of 24LC256 eeprom chip
int a =128;           // Data
int b =255;           // Data
void setup(void)
{
  Serial.begin(9600);
  Wire.begin();
  pinMode(12, OUTPUT);              //Write LED
  pinMode(11, OUTPUT);              //Read LED
  unsigned int address = 0;         // Starting point address 0
  Serial.print("Writing to EEPROM Address ");
  Serial.println(address); Serial.print("Data ");
  Serial.println (a);
  writeEEPROM(disk1, address, a); //Write 128 into location 0
  address = address +1;             //increment the address
  Serial.print("Writing to EEPROM Address ");
  Serial.println (address); Serial.print("Data ");
  Serial.println (b);
  delay (500);                      //Delay to watch the write to eeprom
  writeEEPROM(disk1, address, b);
  address = 0;
  delay(1000);    //  Delay to see Green LED
  Serial.print("Reading from EEPROM Address ");
  Serial.println (address); Serial.print("Data ");
  Serial.println(readEEPROM(disk1, address), DEC);
  address = address +1;
  Serial.print("Reading from EEPROM Address ");
  Serial.println (address); Serial.print("Data ");
  Serial.print(readEEPROM(disk1, address), DEC);
}
```



I2C Project 3 Wiring using eeprom 24LC256

Project 4 -Read Temperature Sensor and Write to eeprom

Now let's tie it all together.

Load Program for Project 4

I2c_EEPROM_Workshop_Project_4

```
------------------------------
Writing to EEPROM Address 0
Data 128
Writing to EEPROM Address 1
Data 255
------------------------------
Reading from EEPROM Address 0
Data 128
Reading from EEPROM Address 1
Data 255
------------------------------
Reading data from TC74
Data 24
Writing to EEPROM Address 0
Data 24
------------------------------
Reading data from TC74
Data 24
Writing to EEPROM Address 1
Data 24
------------------------------
Reading from EEPROM Address 0
Data 24
Reading from EEPROM Address 1
Data 24
```

```
#include <Wire.h>
#define disk1 0x50            // Address of 24LC256 eeprom chip
int temp_address = 73;        // 1001001 written as decimal number TC74A1
int a = 128;                  // Data
int b = 255;                  // Data
unsigned int address = 0;     // Starting point address 0 for eeprom

void setup(void)
{
  Serial.begin(9600);
  Wire.begin();
  pinMode(12, OUTPUT);              // Write 24LC256 LED
  pinMode(11, OUTPUT);              // Read 24LC256 LED
  pinMode(10, OUTPUT);              // Read TC74 LED
  //--------------------------------------------------------------------------
  //       Write variable a and b to eeprom
  //--------------------------------------------------------------------------
 // unsigned int address = 0;                  // Starting point address 0
  Serial.println(" ");
  Serial.println("------------------------------");
  Serial.print("Writing to EEPROM Address ");
  Serial.println(address); Serial.print("Data ");
  Serial.println(a);
  writeEEPROM(disk1, address, a);           //Write 128 into memory location 0
  address = address + 1;                    //increment the address
  Serial.print("Writing to EEPROM Address ");
  Serial.println(address); Serial.print("Data ");
  Serial.println (b);
  writeEEPROM(disk1, address, b);           //Write 255 into memory location 1
  //--------------------------------------------------------------------------
  //    Read eeprom to validate data written
  //--------------------------------------------------------------------------
```

Project 5 Arduino I²C Bus connected to Logic Analyzer

Configuration Logic Analyzer

Analyzers - **I2C**
Configuration - **Edit Settings**
Display - **Decimal**

Connect probe channel 1 to SDA pin A4
Connect probe channel 2 to SCL pin A5
Name channel 1 SDA - Name Channel 2 SCL
Address Display to 7 -bit address bits only