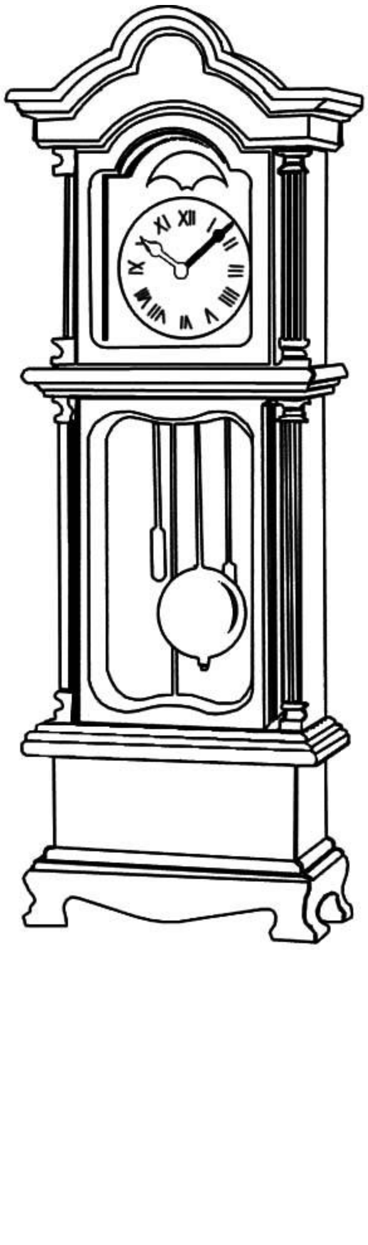
Making an Arduino Real Time Clock



Knowledge is a process of piling up facts; wisdom lies in their simplification.

Martin H Fischer

Modification Record

Issue	Date	Author	Changes (Including the change authority)
A	04 January 2020	ZizWiz	Original from various notes
B			

Contents

1. Introduction

This project attaches a Real Time Clock module (DS3231) to the Arduino Uno. We the add code to the Arduino to display the clock on the LCD screen.

We also write a Windows app that will allow us to set the RTC module to the time of the PC/Laptop.

2. Parts list

For this project we will use:

- Arduino Uno
- DS3231 RTC module
- I^{2}C 4x20 LCD screen

3. Libraries

To make this work a few libraries are needed. Using them makes this a lot simpler.

- DS3231 RTC
  - http://www.rinkydinkelectronics.com/library.php?id=71

- I^{2}C 4x20 LCD screen
  - https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library

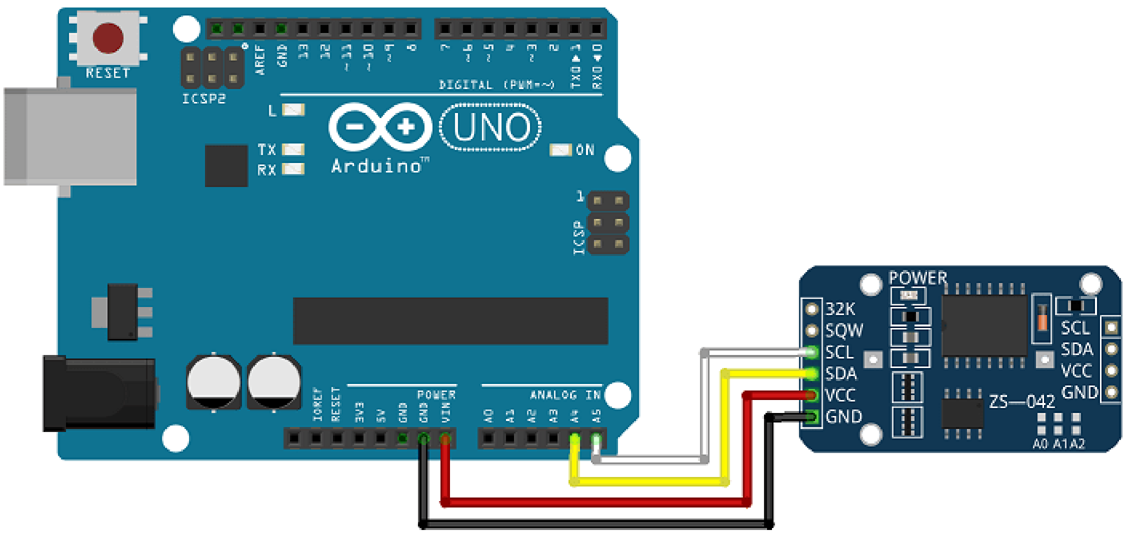
Once installed these are found @ C:\Users\Your_Name\Documents\Arduino\libraries

4. Find Modules Addresses

If you want you can find the address of each correctly wired I^{2}C device on your circuit.

You will find a code sample in Appendix A that you can use to scan for all the I^{2}C devices on your circuit.

5. DS3231 Real Time Clock Module



The DS3231 is a low-cost, highly accurate Real Time Clock which can maintain hours, minutes and seconds, as well as, day, month and year information. Also, it has automatic compensation for leap-years and for months with fewer than 31 days.

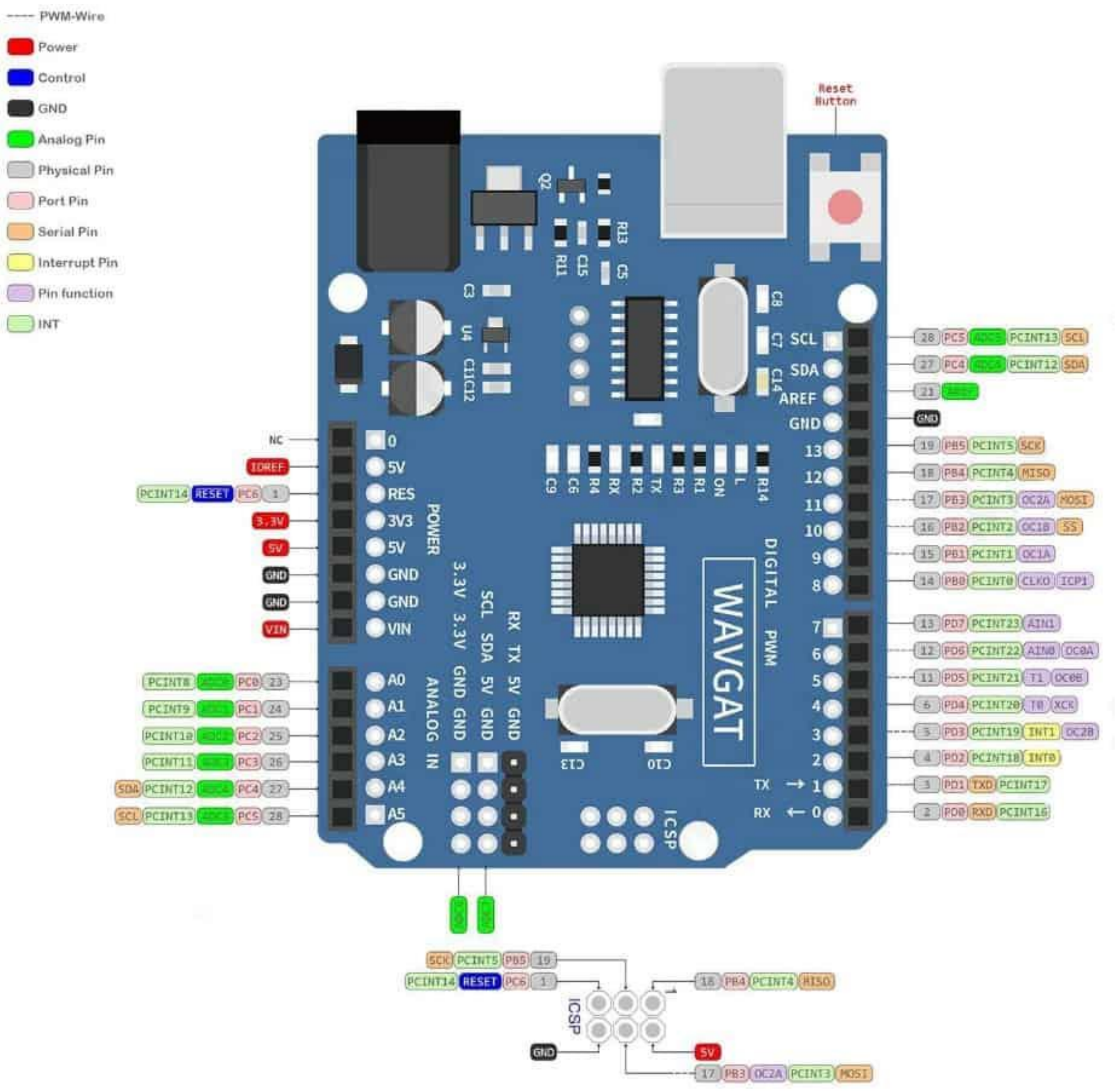
The module can work on either 3.3 or 5 V which makes it suitable for many development platforms or microcontrollers. The battery input is 3V and a typical CR2032 3V battery can power the module and maintain the information for more than a year.

For this example, we put code onto the Arduino to display the time on a 4-line LCD. We set the Clock module time from a windows app, for more details see the explanation later in the document.

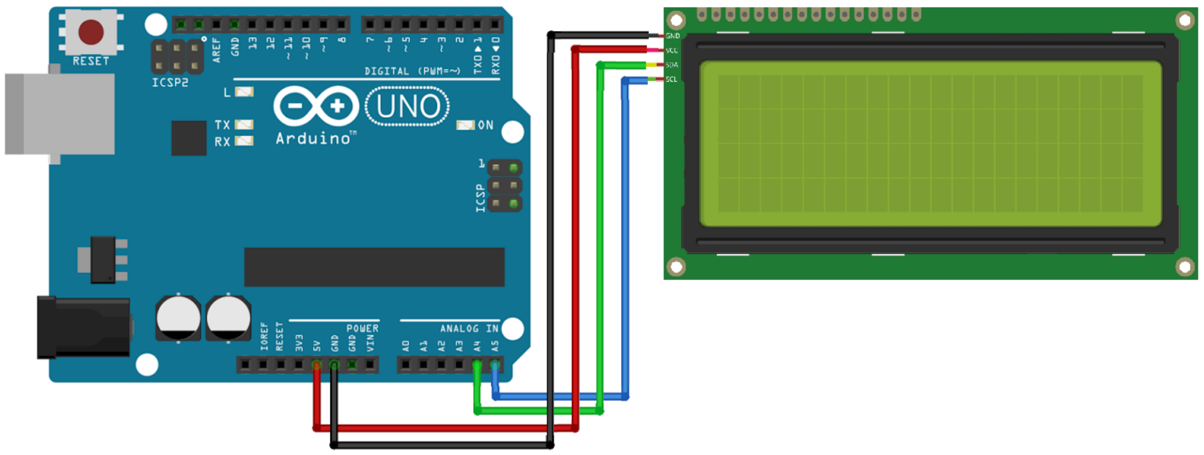
The code to run the clock module is show in Appendix B. The code is commented to explain some finer points of what we do.

6. Arduino Uno

The pinouts for the Arduino are shown below.

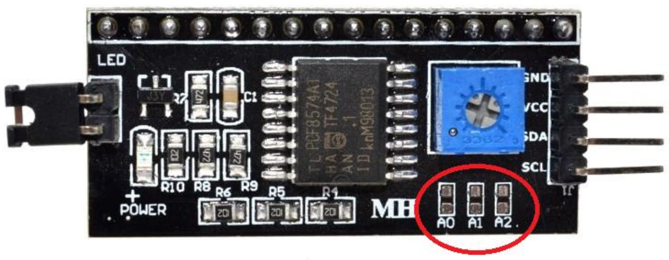


7. I²C LCD



7.1. I²C Address

Some I²C LCD module have an address selector solder pad.

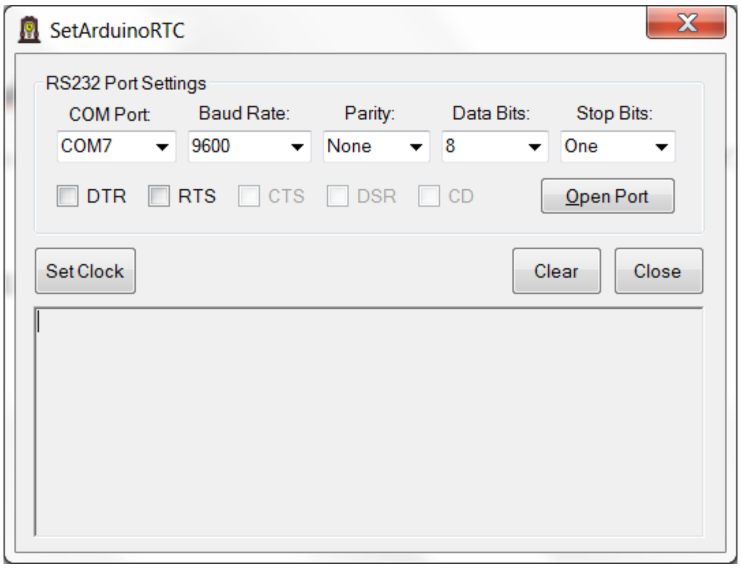


They are usually labelled with A0, A1, and A2. The following table show you how to interpret the selector. "1" = Not Connected, "0" = Connected.

A0	A1	A2	HEX Address
1	1	1	27
0	1	1	26
1	0	1	25
0	0	1	24
1	1	0	23
0	1	0	22
1	0	0	21
0	0	0	20

8. SetArduinoRTC Windows App

The aim of this app is to allow you to set the Arduino clock to the time of your PC/Laptop.


SetArduinoRTC

RS232 Port Settings

COM Port: COM7
Baud Rate: 9600
Parity: None
Data Bits: 8
Stop Bits: One

To set the clock first adjust the port settings then open the port. Now click on Set Clock button and the clock will get set to the PC/Laptop time.

By tickling the DTR line you will be able to reboot the Arduino.

In the Arduino code loop we delay a second between showing the data on the LCD. To compensate for this delay we add a second to the time we send to the Arduino from the PC/Laptop as show in the underlined section below.

```
SendData(ConvertStringToHex(DateTime.Now.ToString("HHmm")) +
ConvertStringToHex(((int)DateTime.Now.Second+1).ToString().PadLeft(2,'0'))
+ date + dow + "04");
```

The C# .Net solutions code is available should you need to adjust it to your needs.

9. Appendix A – Find I²C addresses

```
#include <Wire.h>

void setup() {
  Serial.begin (115200);

  // wait for serial port to connect
  while (!Serial)
    {
    }

  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() {}
```

11. Appendix B – Run Clock

```
/*
 * Setting the time in the Real Time Clock Module
   using DS3231 Library made by Henning Karlsen which can be found and
downloaded from his website, www.rinkydinkelectronics.com.
*/

#include <DS3231.h>
#include <LiquidCrystal.h> // includes the LiquidCrystal Library

String DataString = ""; // a String to hold incoming data
bool Complete = false;  // whether the string is complete

DS3231  rtc(SDA, SCL);
LiquidCrystal lcd(12, 11, 6, 5, 4, 3); //Define LCD display pins
RS,E,D4,D5,D6,D7

void setup() {
  // initialize serial:
  Serial.begin(9600);      //run at 9600
  // reserve 21 bytes for the inputString:
  DataString.reserve(21);

  rtc.begin(); // Initialize the rtc object
  lcd.begin(20, 4); // Initializes the interface to the LCD screen, and
specifies the dimensions (width and height) of the display }

  lcd.setCursor(0, 0);
  lcd.print("Time: ");
  lcd.setCursor(0, 1);
  lcd.print("Date: ");
  lcd.setCursor(0, 2);
  lcd.print("Day: ");
  lcd.setCursor(0, 3);
  lcd.print("Temp: ");
}

void loop() {
  lcd.setCursor(7, 0);
  lcd.print(rtc.getTimeStr());
  lcd.setCursor(7, 1);
  lcd.print(rtc.getDateStr());
  lcd.setCursor(7, 2);
  lcd.print(rtc.getDOWStr());
  lcd.setCursor(7, 3);
  lcd.print(rtc.getTemp());
  lcd.print(char(0xDF));
  lcd.print("C");
  // Set Clock when complete flag has been set:
  if (Complete)
  {
    // Here we set the clock from the incoming data
    rtc.setTime(DataString.substring(0,2).toInt(),
DataString.substring(2,4).toInt(),DataString.substring(4,6).toInt());    //
Set the time e.g. 19:38:00 (24hr format)
    rtc.setDate(DataString.substring(6,8).toInt(),
DataString.substring(8,10).toInt(),DataString.substring(10,14).toInt());
// Set the date DD.MM.YYYY e.g. 04.02.2020
```

```
    clearLCDLine(2); // clear DOW line as they are not all same length
    rtc.setDOW(DataString.substring(14,15).toInt());         // Set Day-of-
Week by number where SUNDAY = 7

    // clear the string and drop flag
    DataString = "";
    Complete = false;

    //send ACK to PC, you can add more checks here like read the data back
to PC as well.
    Serial.write(char(0x06));
  }

  delay(1000);

  // Set Clock when complete flag has been set:
  if (Complete)
  {
    // Here we set the clock from the incoming data
    rtc.setTime(DataString.substring(0,2).toInt(),
DataString.substring(2,4).toInt(),DataString.substring(4,6).toInt());    //
Set the time e.g. 19:38:00 (24hr format)
    rtc.setDate(DataString.substring(6,8).toInt(),
DataString.substring(8,10).toInt(),DataString.substring(10,14).toInt());
// Set the date DD.MM.YYYY e.g. 04.02.2020
    clearLCDLine(2); // clear DOW line as they are not all same length
    rtc.setDOW(DataString.substring(14,15).toInt());         // Set Day-of-
Week by number where SUNDAY = 7

    // clear the string and drop flag
    DataString = "";
    Complete = false;

    //send ACK to PC, you can add more checks here like read the data back
to PC as well.
    Serial.write(char(0x06));
  }
}

/*
  SerialEvent occurs whenever new data comes in the hardware serial RX.
This
  routine is run between each time loop() runs, so using delay inside loop
can
  delay response. Multiple bytes of data may be available. We only send
data
  once.
*/
void serialEvent() {
  while (Serial.available()) {
    // get the new byte:
    char inChar = (char)Serial.read();
    // add it to the DataString:
    DataString += inChar;
    // if the incoming character is EOT 0x04, set a flag so the main loop
can
    // do something with the data:
    if (inChar == 0x04) {
      Complete = true; //raise flag
    }
  }
}

//clear only lines to be rewritten
void clearLCDLine(int line)
{
  lcd.setCursor(7, line);
  for (int n = 7; n < 20; n++) // 20 indicates 0-20 symbols in line.
  {
    lcd.print(" ");
  }
}
```