Arduino Programming for Displaying Frequency

Paul Dionise
MSU ECE 480 Design Team 2, Diamond Team
11/11/15

Abstract: This set of application notes discuss the process of programming an Arduino microcontroller to display input frequencies onto a 7-digit display. The discussion focuses on the steps taken to properly get the display to show the correct output.

Keywords: Arduino, C Programming, 7-Digit Display, Input Library, Port

Introduction:

An Arduino UNO Rev 3 is the Arduino microcontroller used in these application notes. The UNO is a microcontroller board based on the ATmega328P (Arduino). It has 14 digital input/output pins (of which 6 can be used as PAWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button (Arduino). It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC to DC adapter or battery to get it started (Arduino).

The Seven Digit Display used in these applications is a TM1638 LED Display (TM1638). The TM1638 module is designed to be chainable, it uses only one pin for clock, one pin for data, and an additional pin, for strobe, for each module you chain up to 6 (TM1638). That's a total of 8 pins for controlling 48 7-segment digits, 48 buttons and 96 LEDs (48 red/48 green) (TM1638).

Process:

The first step is outputting frequency with the Arduino is to download the FreqMeasure Arduino library. The FreqMeasure Arduino library measures the elapsed time during each cycle an input frequency (FreqMeasure). On the Arduino Uno Rev 3 board, Pin 8 is the frequency input pin (FreqMeasure). After you have downloaded the FreqMeasure library, open the Arduino coding software. In this software there are two things of interest to look at: the void setup() and the void loop(). The void setup initializes values for your code and the void loop runs the code repeatedly (See Figure 1, at end of document).

To use the FreqMeasure library go to File, Examples, and FreqMeasure (See Figure 2). Once clicking on FreqMeasure choose the Serial Monitor Output option. Now you can see the FreqMeasure code (See Figure 3). Using this code you can measure the frequency that enters into the Arduino Frequency Pin 8.

On the Arduino Board you need to make a couple of connections. If you are using the USB connection you can skip to the next part, otherwise you need to connect the VCC pin to 5V and the GND pin to Ground. You also need to connect Pin 8 to whatever source of frequency you want to measure.

Once all the connections are made, you need to upload your code to the Arduino board. To upload the code to the board you first need make sure that you have the proper port selected. Go to tools, then click on port. On Port you will see the Arduino name next to the port that needs to be selected (See Figure 4). Select that port and then you are ready to upload the code to the board. Click on upload and wait for the program to be uploaded to the board.

Now click on Serial Monitor. When the Serial Monitor is open make sure that the Baud rate equals the Serial.begin(Value) you had in your void setup. Now values of frequency should be printing in the serial monitor window (See Figure 5).

Now that the Arduino is printing the values of frequency you can connect it to the 7-Digit Display and show the frequency values there instead of on the Serial Monitor.

Similar to before, you need to download the TM1638 Arduino library and include it into your program by typing: #include <TM1638.h>. Next you need to add the code above the void setup(): TM1638 module(10, 9, 7); These numbers are the pins in the Arduino you connect to the 7-Digit Display. Now in the void loop(): block of code, add the line module.setDisplayToDecNumber(frequency, 0, false). This line of code displays the frequency that was being shown in the Serial Monitor with no leading zeros. In the last line of code in the void loop add a delay so it doesn't display instantaneously. For example delay(1) delays by 1 millisecond.

Now that the code is complete you need to make the connections on the 7-Digit Display. On the 7-Digit Display, connect the input pin 1 (on the top left) to the VCC of the Arduino. Pin 2, connects to the ground of the Arduino. Pin 7 of the Arduino is connected to Pin 5 of the Display. This is a Strobe pin. Pin 9 of Arduino is connected to the CLK of the Display which is Pin 4. Lastly Pin 10 of the Arduino connects to Pin 4 of the Display which is the DIO pin.

Now upload the new program to the board. After it is done uploading the Display should start displaying the same frequency as the Serial Monitor.

Results:

After testing the results using the function generator, it displays the frequency up to 99% accuracy. For example I had an input frequency of 1800 Hz and it output a frequency of 1800.38 Hz which is at 99.97%. I also tested with an 18,000 Hz frequency and it displayed a frequency of 18,007 Hz which is a 99.96% accuracy.

Conclusion:

The Arduino UNO Rev 3 is a very good microcontroller for frequency monitoring because a library already exists to measure frequency. The only piece of code that needs to be changed for the FreqMeasure library is a delay so it doesn't display so many frequencies at a time. The 7-Digit Display portion of the code was also not too complicated to create but it was a bit more difficult since there wasn't as much to go off of. Overall I would recommend the Arduino UNO Rev 3 and the TM1638 Display for anyone who wants to monitor and display frequency.

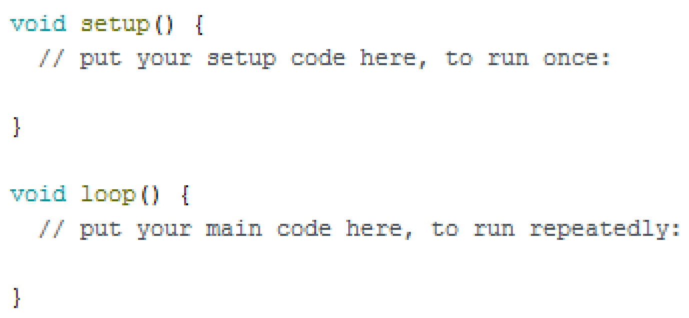
References:

"Arduino - ArduinoBoardUno." *Arduino - ArduinoBoardUno*. N.p., n.d. Web. 12 Nov. 2015.

"TM1638/TM1640 Library." *Reference*. N.p., n.d. Web. 12 Nov. 2015.

"FreqMeasure Library." *, for Measuring Frequencies in the 0.1 to 1000 Hz Range, or RPM Tachometer Applications*. N.p., n.d. Web. 12 Nov. 2015.

Figures:



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

}

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

}
```

Figure 1

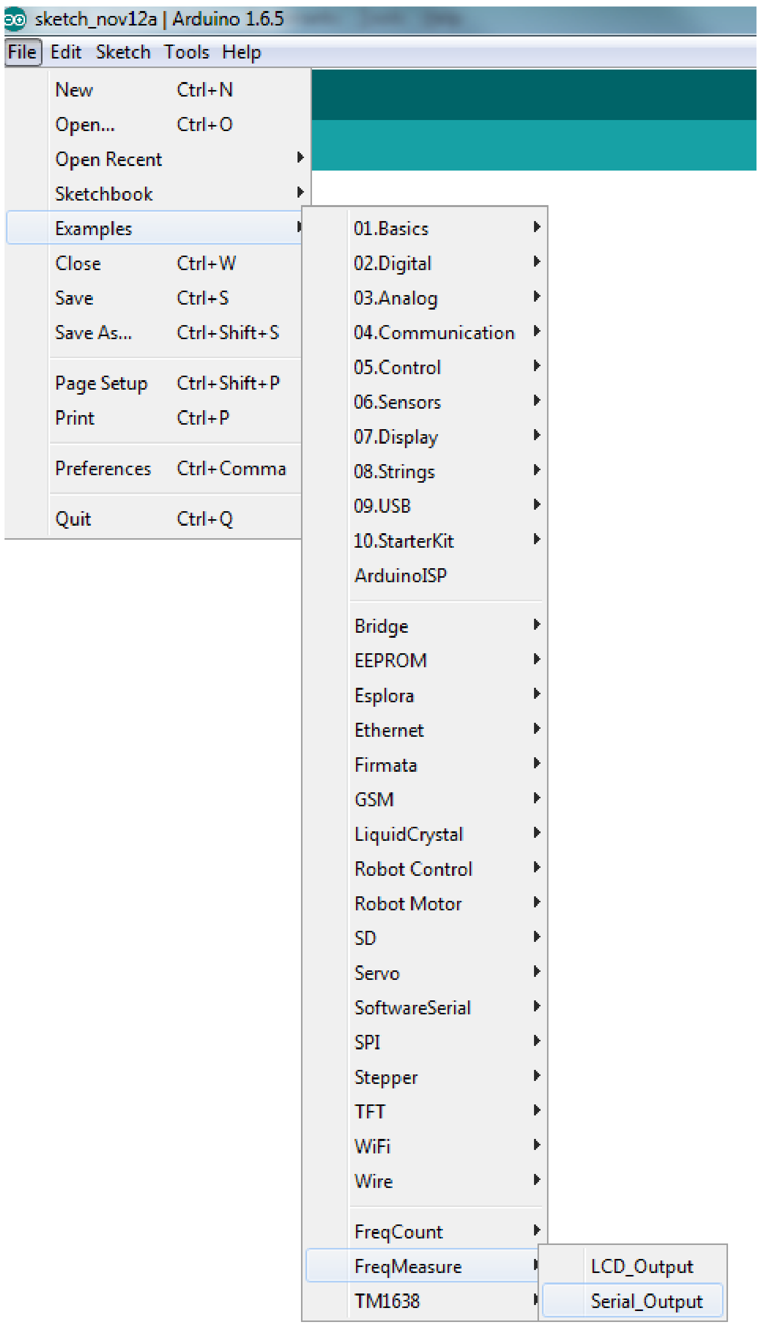
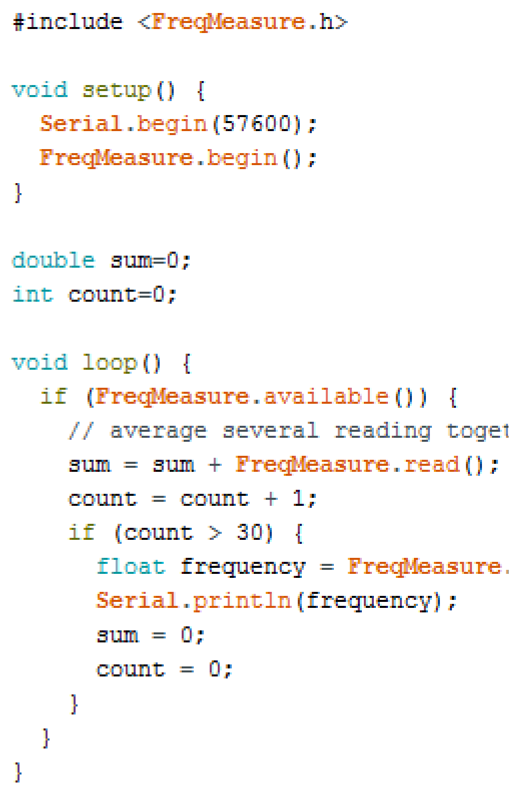


Figure 2



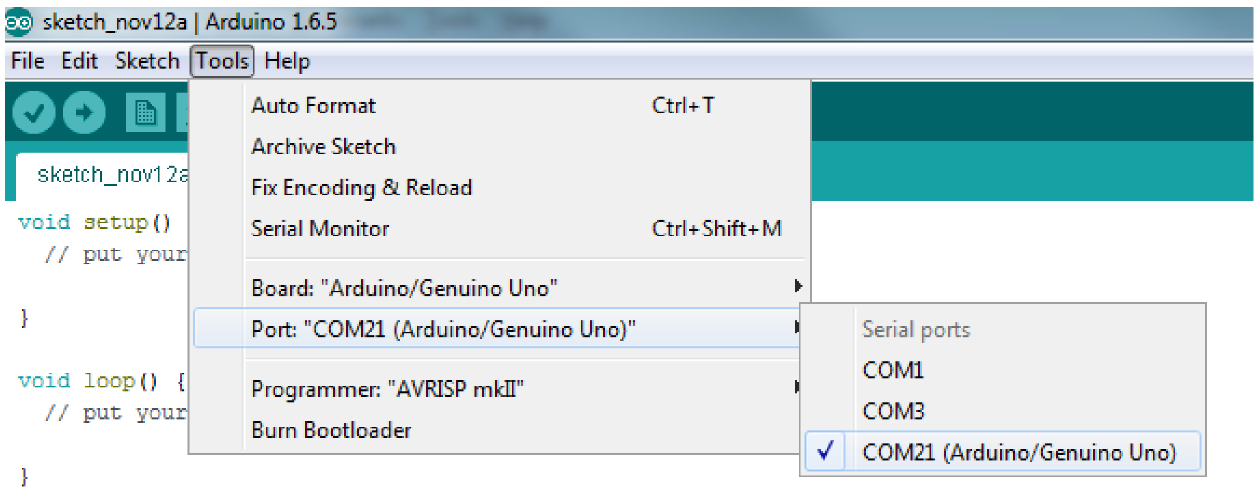
```
#include <FreqMeasure.h>

void setup() {
  Serial.begin(57600);
  FreqMeasure.begin();
}

double sum=0;
int count=0;

void loop() {
  if (FreqMeasure.available()) {
    // average several reading toget
    sum = sum + FreqMeasure.read();
    count = count + 1;
    if (count > 30) {
      float frequency = FreqMeasure.
      Serial.println(frequency);
      sum = 0;
      count = 0;
    }
  }
}
```

Figure 3



sketch_nov12a | Arduino 1.6.5

File
Edit
Sketch
Tools
Help
sketch_nov12a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board: "Arduino/Genuino Uno"
Port: "COM21 (Arduino/Genuino Uno)"
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM1
COM3
✓ COM21 (Arduino/Genuino Uno)

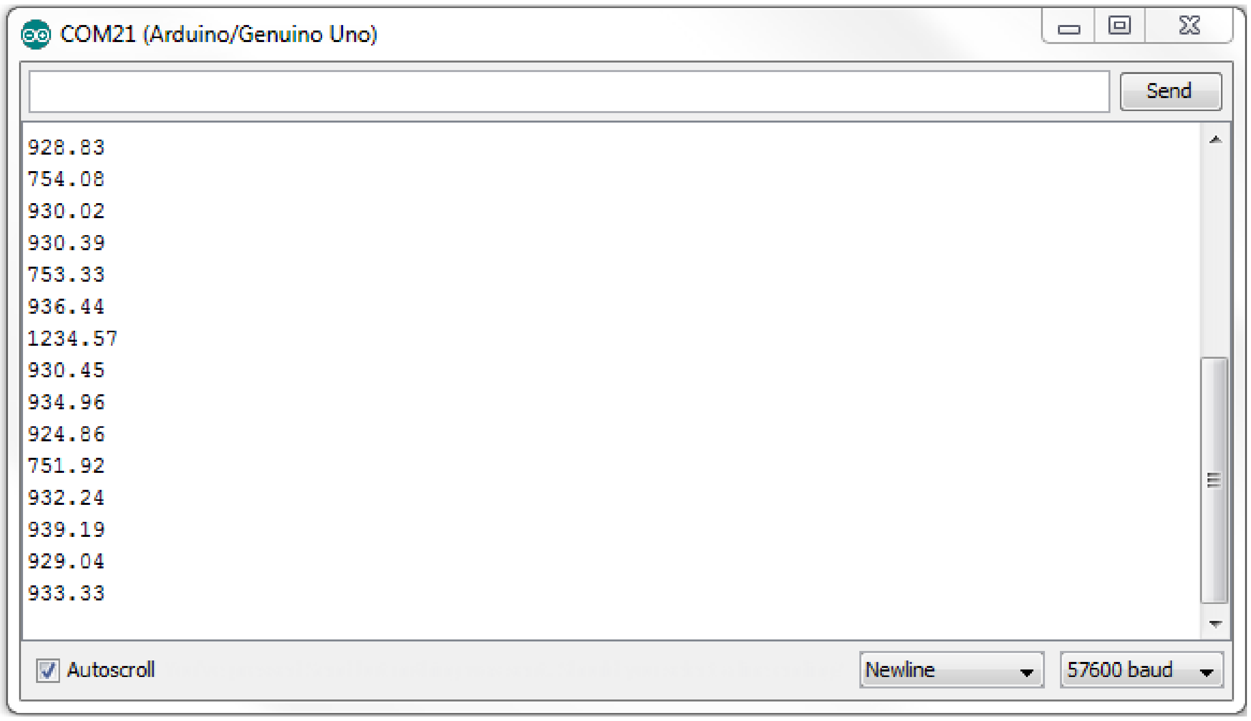
```
void setup()
  // put your

}

void loop() {
  // put your

}
```

Figure 4



```
928.83
754.08
930.02
930.39
753.33
936.44
1234.57
930.45
934.96
924.86
751.92
932.24
939.19
929.04
933.33
```

Autoscroll: ✓
Newline
57600 baud

Figure 5