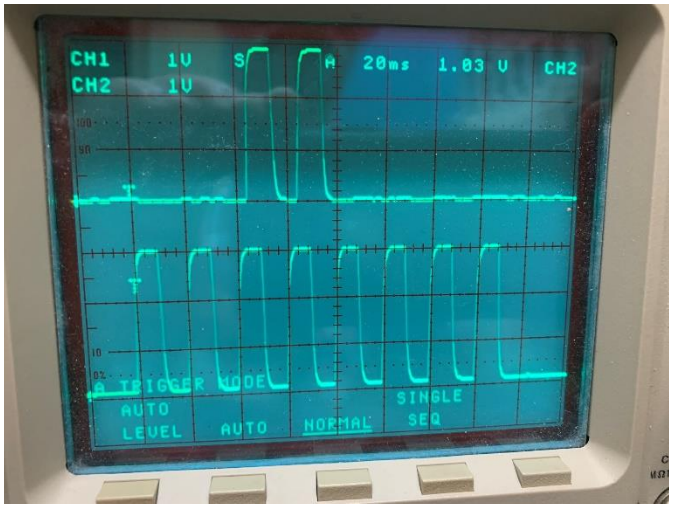
Bridging the Gap Between Realtime and Data in an Arduino, Version 1.0

By R. G. Sparber

Protected by Creative Commons.[1]



I can use my oscilloscope to look at the signal at a given General Purpose Input Output (GPIO) pin. I trust this information as long as its highest frequency does not exceed the capability of the ‘scope.

I can also inject some test code into my program that reads that pin and outputs its state:

```
Sample = digitalRead(LogicalInputPinByte));
Serial.print(Sample);
```

My trust level is not as high because the action of printing the state changes the realtime behavior of the software.

How do I see what the unmodified software sees?

[1] This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by/4.0/ or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

My solution is to write a program that simulates my ‘scope. It triggers when the state of the monitored pin changes state and then takes 255 samples at a user-specified sampling rate. Then it stops recording and dumps the samples to the screen—the cycle repeats.

Here is a typical output:

```
Recorder is armed. + means it triggered: +

Recorded data:
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000
End of recorded data.

Lapsed time was 1 milliseconds.
```

The first line is telling me the program is ready to detect a change in input. When it sees that change, it outputs a “+” and then gets to work reading the input bit and storing it. When 255 samples have been collected, it stops recording and prints out the array. It also records how long it took to record all of this data.

In the above case, I had the sampling rate set to 0. The Lapsed time varies between 1 and 2 milliseconds, which means each sample took between 4 and 8 microseconds. The data varies between all zeros and all ones.

In this next run, I set the sampling rate at 1 millisecond.

```
Recorder is armed. + means it triggered: +

Recorded data:
0000000111111111100000001111111110000000111111111100000001111111
1100000001111111111000000111111111100000001111111111000000111111
1111000000011111111110000001111111111000000011111111110000001111
111111000000011111111110000001111111111000000011111111110000001
End of recorded data.

Lapsed time was 259 milliseconds.
```

The average sampling rate is $\frac{259\, milliseconds}{255\, samples} = 1.02\; milliseconds\; per\; sample$

Notice the pattern `00000001111111111,` which is (7 x 1.02 =) 7.1 milliseconds of zeros followed by 10 .2 milliseconds of ones. The total period is 17.3 milliseconds, which means a frequency of 58 Hz. I'm sure this is 60 Hz, which would be a period of 16.7 milliseconds. Clearly, there is AC noise coupling into my input pin. I just had a clip lead connected to the pin that was stretched across my desk.

By varying the sample rate, I can see fine detail and also get the bigger picture.

The Program

To some, my programming style looks childish or maybe primitive. My goal is to write programs primarily so others can quickly understand what I'm doing. A distance second priority is that the computer can figure it out.

```
//Single channel Almost Realtime Recorder
/***********************************************
This tool lets you see near-realtime changes in an input pin and
have the results dumped to the serial channel.

Trigger occurs when the input pin changes state. Once triggered,
it reads the pin and saves the result in an array as fast as
possible. After NumberOfSamples, it will print readings to the
serial channel. It takes one sample every
SamplingIntervalMsByte milliseconds.
*************************************************
U S E R  D E F I N E D  P A R A M E T E R S
*************************************************/
byte LogicalInputPinByte = 1; //replace number with the logical
input pin you wish to monitor
byte SamplingIntervalMsByte = 1;//sampling rate in milliseconds.
You can specify 0 and it will as fast as it can.
/*************************************************/
unsigned long LapseTimeUlong = 0;
unsigned long EndTimeUlong = 0;
unsigned long StartTimeUlong = 0;
#define NumberOfSamples 255 //255 is the maximum number of
samples (found emperically on Pro Micro)
byte RecordedDataArrayByte[NumberOfSamples];
byte LastReadingByte = 0;
byte SampleNumberByte = 0;

void setup(){
  pinMode(LogicalInputPinByte, INPUT);//set up input pin
  Serial.begin(9600);//set up path to terminal
  delay(1000);
}

void loop(){
     Serial.print(F("Recorder is armed. + means it triggered:
"));
     Trigger();
     ReadDataIn();
     DataOut();
}
```

```
void Trigger(){
     LastReadingByte = digitalRead(LogicalInputPinByte);//read
the input as a baseline and save to detect change
     while(digitalRead(LogicalInputPinByte) ==
LastReadingByte);//add in delay(1) to prevent the watchdog timer
from firing on the ESP8266.
     //when the input changes state, return so data can be read
in
     Serial.println(F("+"));
}


void ReadDataIn(){
     StartTimeUlong = millis();
     if(SamplingIntervalMsByte == 0){
     for (SampleNumberByte = 0; SampleNumberByte <
NumberOfSamples;SampleNumberByte++){
          RecordedDataArrayByte[SampleNumberByte] =
digitalRead(LogicalInputPinByte);
     }
     }else{
          for (SampleNumberByte = 0; SampleNumberByte <
NumberOfSamples;SampleNumberByte++){
          RecordedDataArrayByte[SampleNumberByte] =
digitalRead(LogicalInputPinByte);
          delay(SamplingIntervalMsByte);
          }
     }
     EndTimeUlong = millis();
}


void DataOut(){
     Serial.println();
     Serial.println(F("Recorded data:"));
     //delay(5000);
     for (SampleNumberByte = 0; SampleNumberByte <
NumberOfSamples;SampleNumberByte++){
          byte Sample = RecordedDataArrayByte[SampleNumberByte];
          Serial.print(Sample);
          if((SampleNumberByte+1)%64 == 0)Serial.println();
     }
     Serial.println();
     Serial.println(F("End of recorded data."));
     Serial.println();
     LapseTimeUlong = EndTimeUlong - StartTimeUlong;
     Serial.print(F("Lapsed time was "));
```

```
        Serial.print(LapseTimeUlong);
        Serial.print(F(" milliseconds."));
        Serial.println();
        delay(1000);
}
```

I welcome your comments and questions.

If you wish to be contacted each time I publish an article, email me with “Subscribe” in the subject line. In the body of the email, please tell me if you are interested in metalworking, software, or electronics so I can put you on the best distribution list.

If you are on a list and have had enough, email me “Unsubscribe” in the subject line.

Rick Sparber
Rgsparber.ha@gmail.com
Rick.Sparber.org