Water Meter Reader Documentation

Neptune E-Coder Version

Software Code

Written in C++ Arduino V1.0.2 for Windows

Target Board: Ethernet with USB Serial Light Serial Adapter for programming

Arduino File: DataTimeSDServerFinal_Feb3_14

Release 1.0

February 3, 2014

Reference:Pin Map

Pin 0	Serial Tx
Pin 1	Serial Rx
Pin 2	
Pin 3	
Pin 4	SD card CS (SPI Select)
Pin 5	
Pin 6	Rx Read Meter Data
Pin 7	Tx Clock to Meter
Pin 8	Relay
Pin 9	LED
Pin 10	Ethernet SPI (SS)
Pin 11	Ethernet SPI (MOSI)
Pin 12	Ethernet SPI (MISO)
Pin 13	Ethernet SPI (SCK)
Analog pins 1-6	

Initialization –

Definitions, libraries incuded, Global declarations

Void (setup) –

Initialize Serial Port, SD Card, Ethernet Client, Server.

IP: 192.168.17.176 [May be DHCP from Rozeware Checkbox park intranet server] See intranet setup

Void()

Prog Exec – once every hour

- Engage Relay contact with water meter
- Void(GetTime) Get internet time from UDP NTP
- Void(ReadMeter) Read Water Meter
- Void(SDCardWrite) Write date, time, Water Meter Reading to SD Card
- Disengage Relay contact with water meter
- Monitor Client request for data
    - Send Requested SD Data

```
GetTime()

Disable SD
Enable Ethernet
Set UDP listen port to 8888
Set Ethernet client as DNS Client
Request Time Server IP address

Create NTP packet request
Send NTP request to DNS Server (pool.ntp.org)
Get UDP packet from assigned  DNS  server
Decode NTP  time
        -   Add day to Microsoft date code
        -   Decode hour and minute
```

```
ReadMeter()   *See Appendix

Put clock on Tx line until Rx line goes high (<10 Min)
Record bits when Rx line goes low (62 – 95mS)
Read 34 Data Bytes
  -   11 bit bytes. (1 start, 8 data, 2 stop)
  -   8 samples/bit read sample 5
Timing very critical. Should be hardware decoder not software
Or use edge detect (Arduinos  edge detect voids delay timing)
  -   Align data bits and strip out digit
```

```
SDCardWrite()


Disable Ethernet
Enable SD Card
Create String
        -   Write date, time, data (bytes 7-12)
        -   Add 24 Hour marker
        -   Add EOF (Null)
Open SD
Write SD
Close SD
Disable SD
Erase String
```

Client Monitor

Client Requests data from Server

- $1 = All Data
- $2 = All days one reading per day
- $3 = Last 32 hours
- $4 = All Data last 7 Days
- $5 = All Data last 31 days
- $6 = Once per day last 31 days

Respond to establish handshake with content type

Open SD data file DateData.TXT

Write data to Ethernet

Close SD Card

Disconnect Client

Code on Arduino Ethernet Card

Arduino File: DataTimeSDServerFinal_Feb3_14

```
/*

 Web Server, Web client, water meter reader, MicroSD card

File:  DateData.txt

Circuit: Ethernet shield attached to pins 10, 11, 12, 13

SD Enable attached to pin 4

TxClk pin 7

RxData pin 6

led pin 9

Relay pin 8


*/
```

```
#include <SPI.h>
#include <Ethernet.h>
#include <EthernetUdp.h>
#include <Dns.h>
#include <SD.h>

#define RxData 6
#define TxClock 7
#define Relay 8
///////////////////Ethernet Setup
// Enter a MAC address for your controller below.
// Newer Ethernet shields have a MAC address printed on a sticker on the shield
byte mac[] = {   0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = {   192,168,17,176 };   // ip in LAN
unsigned int localPort = 8888;      // local port to listen for UDP packets

IPAddress timeServer; // pool.ntp.org NTP server

const int NTP_PACKET_SIZE= 48; // NTP time stamp is in the first 48 bytes of the message
byte packetBuffer[ NTP_PACKET_SIZE]; //buffer to hold incoming and outgoing packets

EthernetServer server(80);
EthernetClient client;
EthernetUDP Udp;  // A UDP instance to let us send and receive packets over UDP
```

```
/////////////////////////////Read Meter declarations

unsigned int dataAlign[35];              // 35 is ok Buffer for bit read data

unsigned int meterByte[35];              // 35 is ok


int count = 9;                  //byte timing tuning

int bitcount =0;

int mask = 15;              // mask 0b0000 0000 0000 1111. Strips 4 bit integer

unsigned int last=0;       // Send Data record count

unsigned int last_A=0;     // Send Data count register


byte set_P=0;       // Send Data start / stop flag

byte set=0;       // command flag in ethernet read

byte command =0;  // Data command


unsigned long previousMillis = 0;// Delay between  Meter  reads


// added 1.25 seconds to hourly read - Feb 1st coldest month of year - to correct for time slippage

unsigned long interval = 3601250; //1 Hour Time delay between NTP reads(3,600,000)


unsigned int Current_Date = 0;

unsigned int Current_Hour = 0;

unsigned int Current_Minute = 0;

int bitRate=415;                     //  1187hz. Seems more stable than 1200

boolean state=false;
boolean laststate = false;





void setup() {
  Serial.begin(9600);

  pinMode (RxData,INPUT_PULLUP);//  Read
  pinMode (TxClock,OUTPUT);// clock
  pinMode (Relay,OUTPUT);

  // disable w5100 while setting up SD
  pinMode(10,OUTPUT);
  digitalWrite(10,HIGH);

  Ethernet.begin(mac, ip);
  digitalWrite(10,HIGH);
    server.begin();
}
```

```
void loop() {
 unsigned long currentMillis = millis();


 if (currentMillis - previousMillis > interval) {
  previousMillis = currentMillis;
  digitalWrite(Relay,HIGH);
  delay (200);
  GetTime();                                 // call GetTime function
  MeterRead();                               // call get meter data
  SDcardWrite();                             // call SD write


  return;                   //seem to need to re initialize?
 }
```

// Create a client connection

```
  EthernetClient client = server.available();
 if (client) {
  while (client.connected()) {
   if (client.available()) {
    char c = client.read();
     if(set==1){          // set by receiving  $ char
     command = c;    // char following the $ char defines what to send
     set=0;
    }
```

```
if(c==36) set=1;        // char $ recieved



if (c == '\n') {                        //  Null =  HTTP request has ended
  client.println(F("HTTP/1.1 200 OK")); //send new page
  client.println(F("Content-Type: text"));
  client.println();

  SD.begin(4);
  File myFile = SD.open("DateData.TXT");
  delay(1);
  if (myFile) {

    while (myFile.available()) {
      char c=(myFile.read());
      if (c == 13)  last++;    // use CR as record increment count

      // All Data client.print command
      if(command==49)  client.write(c); // command =1

      // 1 per day All data client.print command
      if((command==50)&&(c == 'h')) set_P=0;
      if((command==50)&&(set_P == 1))  client.write(c);
      if((command==50)&&(c == 'H')) set_P=1;   // 1 per day all days  ( Written in SD string)
      // All data last 32 hours client print command

      if((command==51) && (last > (last_A - 33))) client.write(c); //send last 32



      // All data last 7 days client print command

      if((command==52) && (last > (last_A - 169))) client.write(c); //send last 7 days


      // All data 31 days client print command

      if((command==53) && (last > (last_A - 745))) client.write(c); //send last 24


      // 1 per day last 31 days at 23 hour

      if((command==54)&&(c == 'h')) set_P=0;

      if((command==54)&&(set_P == 1))  client.write(c);

      if((command==54) &&  (last > (last_A - 745)) && (c == 'H')) set_P=1 ;

    }

    myFile.close();

  }

  delay(1);

  client.stop();

  digitalWrite(4,HIGH);

  delay(1);


  last_A=last; // cumulative SD record count (Also incremented in SD Write)

  last=0;     // clear 'last' record count register
```

```
     command=0;    // clear 'command' register

    }
   }
  }
 }
}
void MeterRead()
{
 digitalWrite(TxClock,HIGH); // set up to put an initial low on clk line
 digitalWrite(Relay,HIGH);
 delay (200);

 //Clk until Rx line changes (Up to 10 minutes)
 state = digitalRead(RxData);
 while (state == digitalRead(RxData))
 {
  digitalWrite(TxClock,HIGH); //inverted due to transistor
  delayMicroseconds(bitRate);
  if(state !=digitalRead(RxData)) break;
  digitalWrite(TxClock,LOW);
  delayMicroseconds(bitRate);
 }
```

// Look for Rx line to go Low (62 - 95mS)

```
//Quickly align transistion of state change
  state = digitalRead(RxData);
while (state == digitalRead(RxData)) {
  for( int y=0; y<32; y++) {
    if (y==0) digitalWrite(TxClock, HIGH);
    if (y==15) digitalWrite(TxClock, LOW);
    delayMicroseconds(20);
    if(state != digitalRead(RxData)) break;
    }
}
```

// Read 34 Data bytes. 316mS to 319mS per 34 bytes read.

```
for(int mData=0; mData < 34; mData++){
  for (int bytecount=0; bytecount < 11; bytecount++) {    // 11 bits per byte incl. 2 stop and 1 start
    for(bitcount =7; bitcount >=0; bitcount--){     //   read each bit 8 times. 4 high, 4 low
                    if (bitcount == 7) digitalWrite(TxClock, HIGH);
                    if (bitcount == 3) digitalWrite(TxClock, LOW);
      delayMicroseconds(96);                                    // 1180 bits/Sec. 107 bytes/Sec
      laststate =! digitalRead(RxData);
      if (bitcount==5)
        bitWrite(meterByte[mData], bytecount, laststate);// write bit state
```

```
   }

   delayMicroseconds(count); // fine tune timing. Count from 7 to 11

  }
  //dataAlign[mData] = meterByte[mData];  //align 11 bit bytes
  if (bitRead(meterByte[mData], 10)==0) dataAlign[mData] = meterByte[mData];  //should be start bit
  if (bitRead(meterByte[mData], 10) >0) dataAlign[mData] = meterByte[mData] >>1;  //shift right may correct align
  if ((bitRead(meterByte[mData], 5) >0) && (bitRead (meterByte[mData], 6) >0)) dataAlign[mData] = meterByte[mData] >>1;  //bit 5 & 6 should be 1
  if ((bitRead(meterByte[mData], 6) >0) && (bitRead (meterByte[mData], 7) >0)) dataAlign[mData] = meterByte[mData] >>2;  //align 11 bit bytes

  meterByte[mData] = dataAlign[mData] & mask; // meterData is least 4 bits of masked aligned data

~~  // delayMicroseconds(count); // maybe more fine timing tuning~~
 }

 //exit read loop

 digitalWrite(TxClock,HIGH); // put low on meter

 digitalWrite(Relay,LOW); //Turn off relay before writing to SD card

}
//end data capture
```

```
//Write data to SDCardFile

void SDcardWrite() {

  pinMode(10,OUTPUT);          // disable w5100 SPI while starting SD
  digitalWrite(10,HIGH);
  SD.begin(4);

  String dataString = "";                    // clear string
  dataString += Current_Date;
  //dataString += ",";
  dataString += "\t";                         //TAB
  dataString += Current_Hour;
  dataString += ":";
  if (Current_Minute < 10 ) {    // In the first 10 minutes of each hour, we'll want a leading '0'
    dataString += "0";
  }
  dataString += Current_Minute;
  //dataString += ",";
  dataString += "\t";                          //TAB
  dataString += meterByte[7];
  dataString += meterByte[8];
  dataString += meterByte[9];
  dataString += meterByte[10];
  dataString += meterByte[11];
  dataString += (".");
  dataString += meterByte[12];

  if (Current_Hour==23){                    //Write Hour marker
    dataString += "\t";
    dataString += "H";
  }
  else{
    dataString += "\t";
    dataString += "h";
  }

  dataString += "\0"; // null

   // write to SD file
  File dataFile = SD.open("DateData.txt", FILE_WRITE);
  if (dataFile) {
    dataFile.println(dataString);
    dataFile.close();
  }
  dataString = ""; //delete string
  digitalWrite(4,HIGH);     // disable SD SPI
  last_A++;  // increment record count
```

```
}

void GetTime(){

  pinMode(4,OUTPUT);    // disable SD SPI while starting w5100
 digitalWrite(4,HIGH);

 pinMode(10,OUTPUT);
 digitalWrite(10,HIGH);



 Udp.begin(localPort);
 DNSClient dns;
 dns.begin(Ethernet.dnsServerIP());
 if(dns.getHostByName("pool.ntp.org",timeServer))
  sendNTPpacket(timeServer);          // send an NTP packet to a time server

  // wait to see if a reply is available
 delay(4000); // was 1000
 if ( Udp.parsePacket() ) {      // We've received a packet, read the data from it
  Udp.read(packetBuffer,NTP_PACKET_SIZE);  // read the packet into the buffer

  unsigned long highWord = word(packetBuffer[40], packetBuffer[41]);
  unsigned long lowWord = word(packetBuffer[42], packetBuffer[43]);
```

```
    unsigned long secsSince1900 = highWord << 16 | lowWord;

    const unsigned long seventyYears = 2208988800UL;
    unsigned long epoch = secsSince1900 - seventyYears;

    //Current_Date = (41244 +((epoch - 1354341600) / 86400));// DST
    Current_Date = (41244 +((epoch - 1354338000) / 86400));// EST

    //Current_Hour = (((epoch - 14400)  % 86400L) / 3600); // DST -4 Hrs from UTC
    Current_Hour = (((epoch - 18000)  % 86400L) / 3600); // EST -5 Hrs from UTC

    Current_Minute = ((epoch  % 3600) / 60); // print the minute (3600 equals secs per minute)

  }
}

// send an NTP request to the time server at the given address
unsigned long sendNTPpacket(IPAddress& address)
{
  memset(packetBuffer, 0, NTP_PACKET_SIZE); // set all bytes in the buffer to 0
  packetBuffer[0] = 0b11100011;   // LI, Version, Mode
  packetBuffer[1] = 0;     // Stratum, or type of clock
  packetBuffer[2] = 6;     // Polling Interval
  packetBuffer[3] = 0xEC;  // Peer Clock Precision
  // 8 bytes of zero for Root Delay & Root Dispersion
  packetBuffer[12]  = 49;
  packetBuffer[13]  = 0x4E;
  packetBuffer[14]  = 49;
  packetBuffer[15]  = 52;

  // all NTP fields have been given values, now
  // you can send a packet requesting a timestamp:
  Udp.beginPacket(address, 123); //NTP requests are to port 123
  Udp.write(packetBuffer,NTP_PACKET_SIZE);
  Udp.endPacket();

}
```

See below.

Appendix:

Bit 1: STX (2)

Bit 2: Some sort of Indicator. Either a 1 or a 2. Affects bits 26**, 28, 29**, 30

Bit 3 – 6: Always 0037

Bit 7: ETB (7)

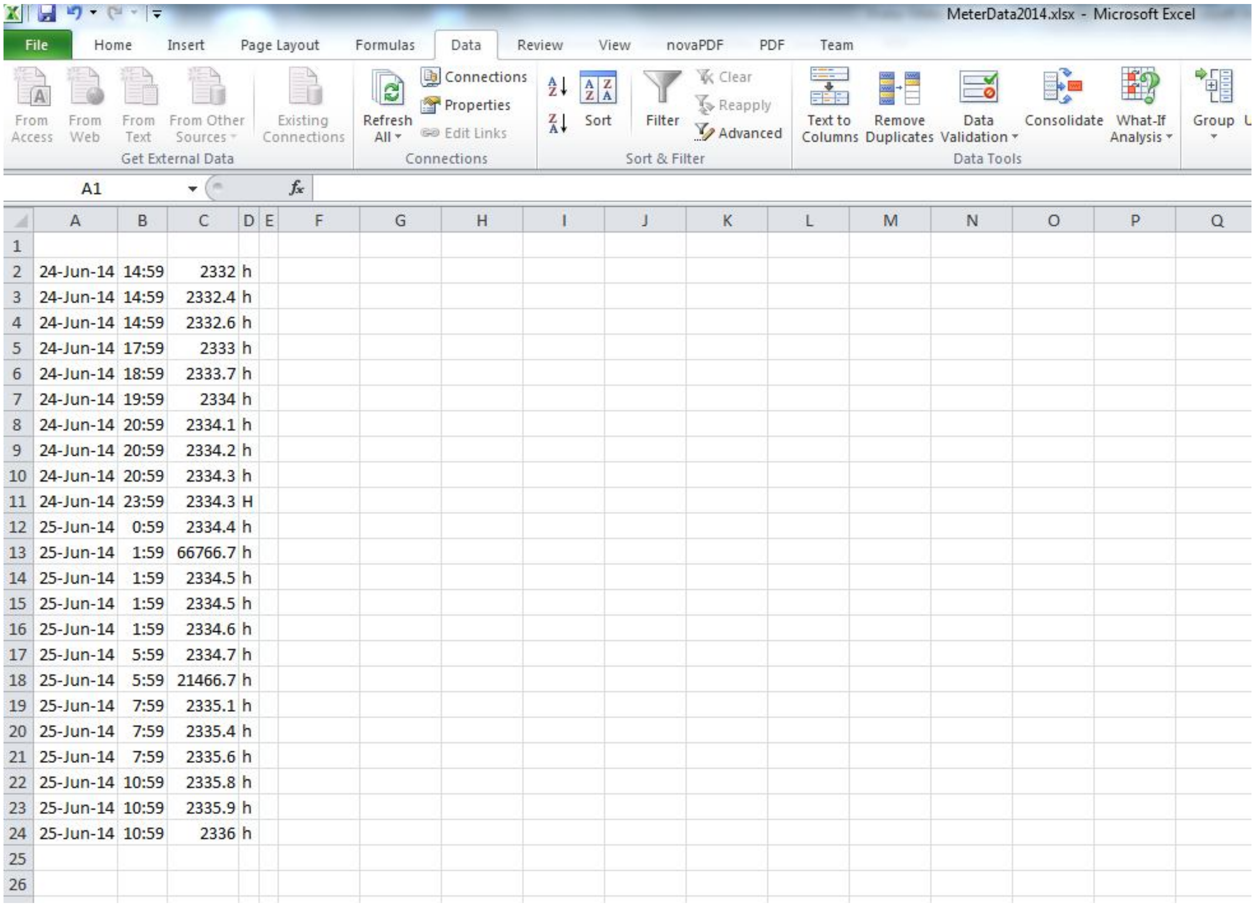
Bit 8 – 13: Data (Cubic Meters. (00000.0)

Bits 15 – 24:   Meter Number: 9122130023

Bits 26 -34: Additional data: probably X.X**00** data. 28, 29 are always 10, 7 if bit 2 is a 1. 26 changes to 8.

Red   reads are bad reads

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
		8	9	10	11	7	8	9	10	11	7	8	9	10	11	7	8	9	10	11	7	8	9	10	11
		4:52	5:20	5:47	6:17	6:46	7:30	8:35	9:14	9:50	10:22	10:55	4:52	5:36	6:22	8:12	8:56	9:34	10:13	10:49	11:26	11:56	12:32	12:56	14:22
1	STX(2)	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	2	1	1	1	1	2	1	2	1	2	1	1	1	2	1	1	1	1	1	1	2	1	2	1	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	S(3)	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
6	W(7)	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
7	ETB(7)	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
11	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
12	6	6	6	6	6	6	6	6	6	6	6	6	6	7	7	7	7	7	7	7	7	7	7	7	7
13	5	9	9	9	9	9	9	9	9	9	9	9	9	0	0	0	0	0	0	0	0	1	1	1	1
14	ETB(7)	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
15	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
17	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
18	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
19	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
20	3	3	3	3	11	3	3	3	3	3	3	3	3	3	1	11	3	3	3	3	11	3	3	3	3
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	3	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	3	0	0	0	0
23	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	9	2	2	2	2	9	2	2	2	2
24	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	11	3	3	3	3	11	3	3	3	3
25	ETB(7)	7	7	7	7	7	7	7	7	7	14	7	7	7	14	11	7	7	7	7	11	7	7	7	7
26	(space) (0)	0	0	0	0	8	0	8	0	8	0	0	0	8	0	1	0	0	0	0	1	0	8	0	0
27	ETB(7)	11	7	7	7	7	7	7	7	7	14	7	7	7	14	11	7	7	7	7	11	7	7	7	7
28	1	9	10	10	10	4	10	4	10	5	4	10	10	2	4	9	10	10	10	10	2	10	0	10	10
29	6	13	7	7	7	1	7	1	7	2	14	7	7	5	14	13	7	7	7	7	3	7	4	7	7
30	@(0)	1	12	12	12	0	1	0	12	0	8	12	12	0	8	1	12	12	12	12	1	12	0	12	12
31	ETB(7)	11	7	7	7	7	11	7	7	7	14	7	7	7	14	11	14	7	7	7	11	7	7	7	7
32	7	5	5	5	5	6	5	6	5	7	10	5	5	5	6	3	0	6	6	6	13	5	6	5	5
33	2	3	2	2	4	4	3	4	2	6	4	14	2	5	3	11	6	3	3	3	3	4	3	4	4
34	ETX(3)	11	6	6	0	3	11	6	6	3	13	11	6	6	0	11	9	6	6	0	11	13	6	6	6



	A	B	C	D
1				
2	24-Jun-14	14:59	2332	h
3	24-Jun-14	14:59	2332.4	h
4	24-Jun-14	14:59	2332.6	h
5	24-Jun-14	17:59	2333	h
6	24-Jun-14	18:59	2333.7	h
7	24-Jun-14	19:59	2334	h
8	24-Jun-14	20:59	2334.1	h
9	24-Jun-14	20:59	2334.2	h
10	24-Jun-14	20:59	2334.3	h
11	24-Jun-14	23:59	2334.3	H
12	25-Jun-14	0:59	2334.4	h
13	25-Jun-14	1:59	66766.7	h
14	25-Jun-14	1:59	2334.5	h
15	25-Jun-14	1:59	2334.5	h
16	25-Jun-14	1:59	2334.6	h
17	25-Jun-14	5:59	2334.7	h
18	25-Jun-14	5:59	21466.7	h
19	25-Jun-14	7:59	2335.1	h
20	25-Jun-14	7:59	2335.4	h
21	25-Jun-14	7:59	2335.6	h
22	25-Jun-14	10:59	2335.8	h
23	25-Jun-14	10:59	2335.9	h
24	25-Jun-14	10:59	2336	h
25				
26				

This is a representation of acquired data in a spreadsheet. Note a couple of bad reads.

Could do better with a hardware implementation for data edge detect instead of Arduino.

Ran out of time to do it correctly, but it works.