HW3 Calibrated Controlled Propeller

Johns Hopkins University

Real Time Software for Embedded Systems

Fall 2014

Tony Florida

2014-10-07

Requirements

Hardware

- There shall be at least one IR led emitter and detector circuit
- There shall be two motor that are spun via electronic speed control (ESC)
- An Arduino board shall control the circuits
- External battery supply shall be used for the motors
- Blades shall be attached to at least one motor and pass through an emitter/detector pair

Software

- The software running on the Arduino shall use function queue scheduling design
- The software shall capture the rotations per minute (RPM) every second
- The software shall capture the command every second
- The software shall capture the time every second
- The software shall command the motors via the ESC from min rotation speed to max, then back down to min

Parts List

- (1) Arduino Uno
- (1) 10k resistor
- (1) 220 ohm resistor
- (1) spool of hobby wire
- (1) USB 2.0 A/B cable
- (1) breadboard
- (1) 12v battery preferable Zippy Compact 25c Series 4000 Li-Po Battery
- (1) XT60 Connector Pair
- (1) LED of any color
- (1) Infrared Emitter and Detector (Radio shack 276-142)
- (2) Turnigy Multistar 30 Amp Multi-rotor Brushless ESC 2-4S
- (1) Gemfan 9x4.7 Nylon Prop Set (1x CW & 1x CCW)
- (2) Turnigy Multistar ESC Programming Card
- Wood to hold motor and emitter/detector
- Nails
- Twist ties
- Black electrical tape

Required Software

- Arduino Sketch v1.0
- DuinOS v0.4
- AVRQueue
- Microsoft Excel 2010

Architecture

Hardware

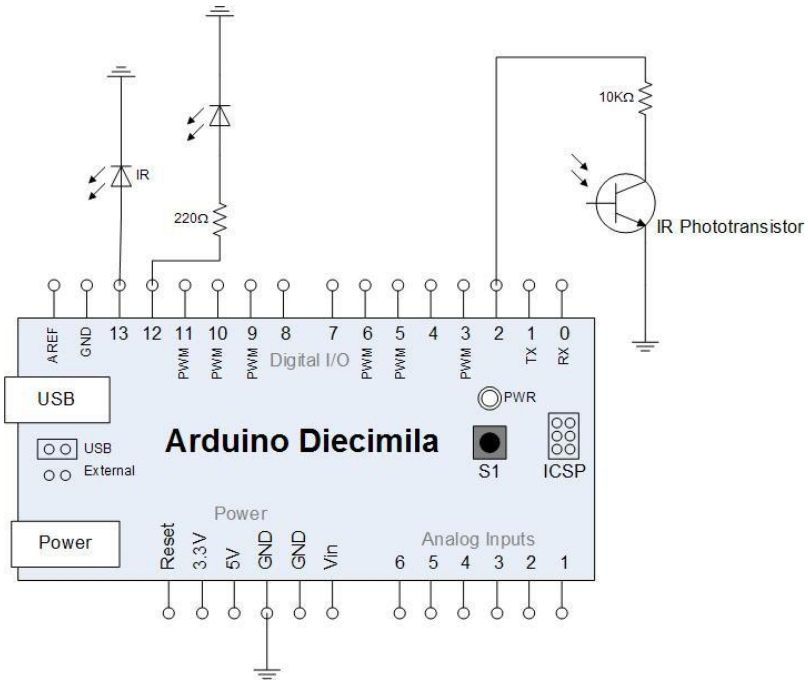


Figure 1 - Circuit Schematic [1]

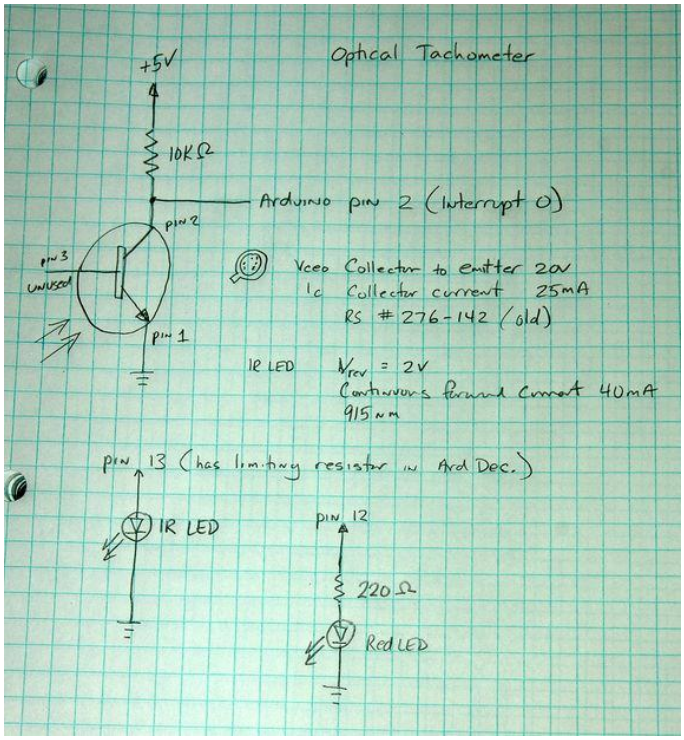


Figure 2 – Schematic Notes [1]

Figure 3 – Emitter Detector Sensors

Software

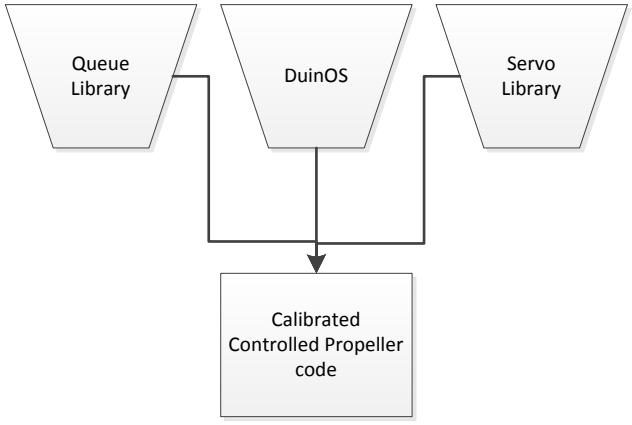


Figure 4 - Software Architecture Diagram

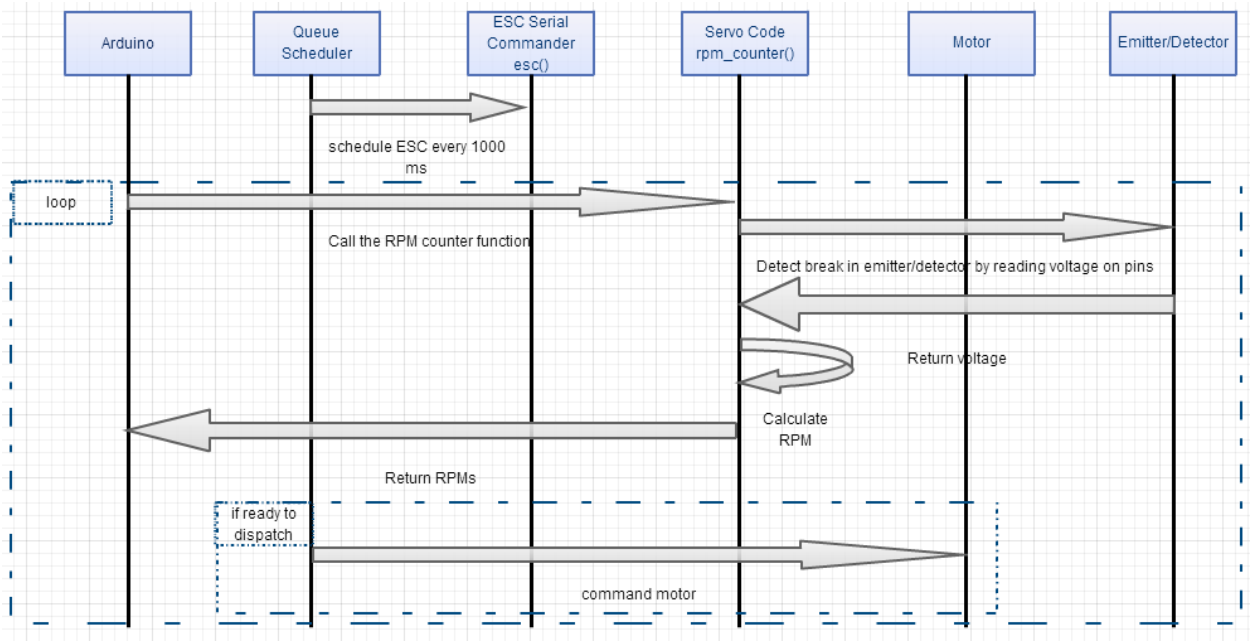
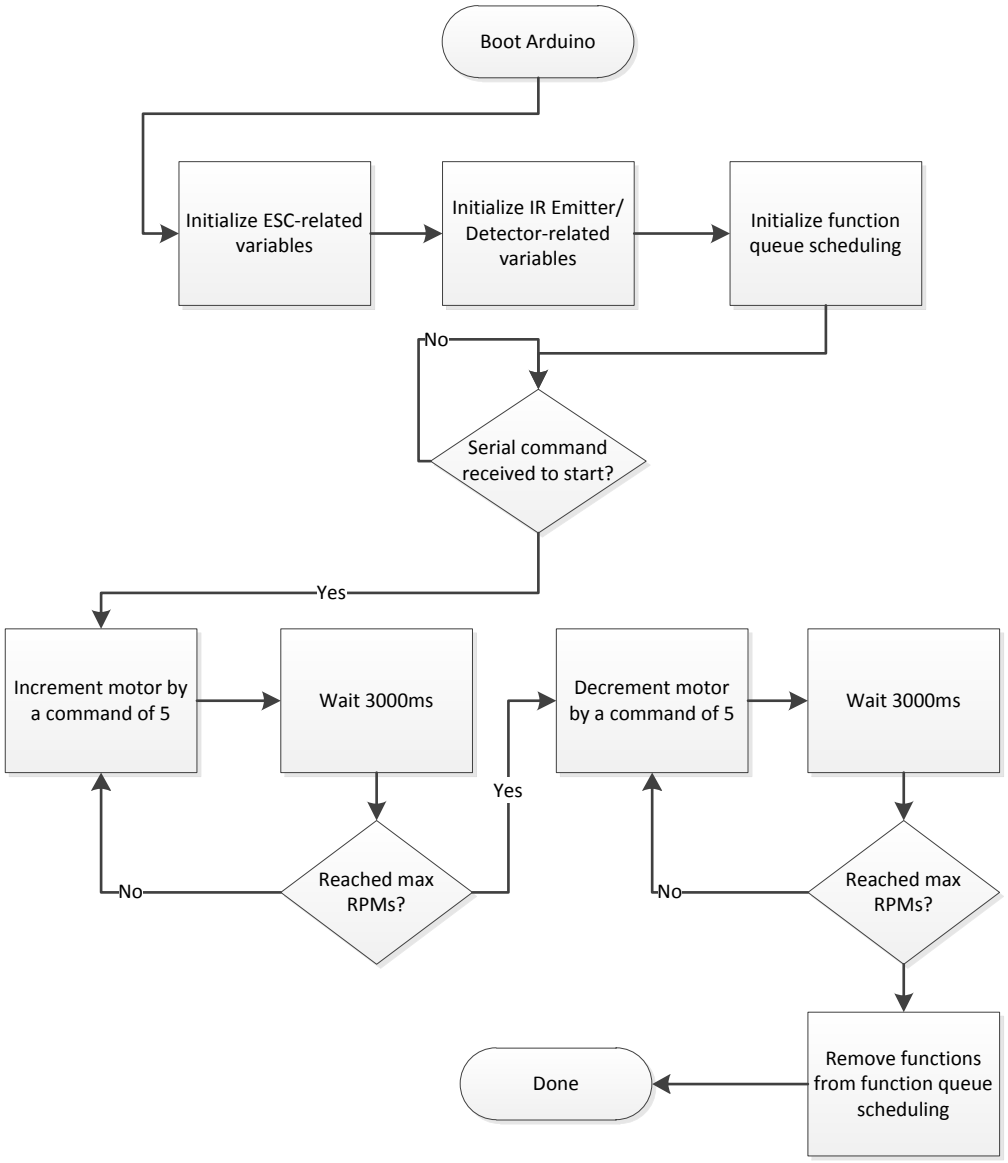


Figure 5 - Hardware/Software Sequence Diagram

Design

Software



Photos of the Hardware

Figure 6 – Tactical Test Setup

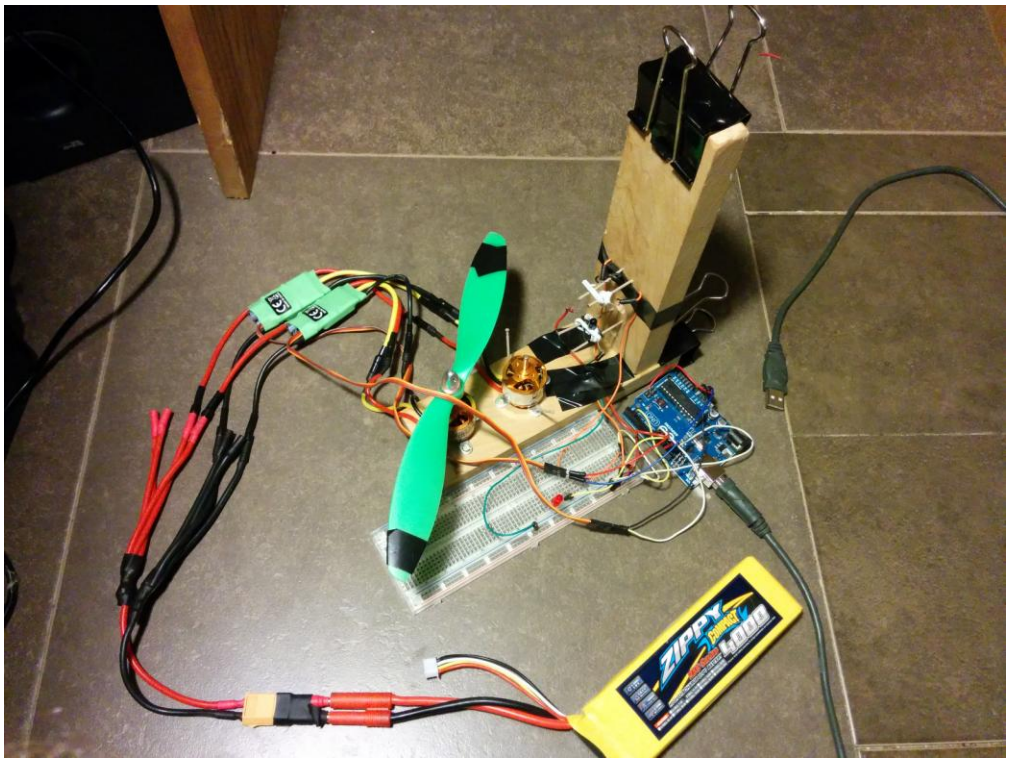


Figure 7 - Top View of the Test Setup

Implementation

```
#include <Queue.h>
#include <Servo.h>

//JHU RTSW HW 3 - HW3 - Calibrated Controlled Propeller
//Tony Florida
//2014-10-06
//References:
// http://www.instructables.com/id/Arduino-Based-Optical-Tachometer/
// http://techvalleyprojects.blogspot.com/2012/06/arduino-control-
escmotor-tutorial.html
// https://github.com/Zuph/AVRQueue

//CMD variables (range of 15 to 150)
int serial_cmd = 5; //serial commands start at 10
int MAX_SERIAL_CMD = 100; //max serial command
int SERIAL_CMD_INCREMENT = 5; //increment serial commands by 20

//Queue variables
Queue myQueue;

//ESC variables
// This is our motor.
Servo myMotor;
Servo myMotor2;
// This is the final output
// written to the motor.
String incomingString;

//IR Emitter Detector variables
int ledPin = 13;                // IR LED connected to digital pin 13
int statusPin = 12;             // LED connected to digital pin 12
volatile byte rpmcount;
volatile int status;
unsigned int rpm;
unsigned long timeold;

//IR Emitter Detector function
void rpm_fun()
{
   //Each rotation, this interrupt function is run twice, so take that
into consideration for
   //calculating RPM
   //Update count
      rpmcount++;

   //Toggle status LED
   if (status == LOW) {
     status = HIGH;
   } else {
```

```
      status = LOW;
    }
    digitalWrite(statusPin, status);
}

int cmd()
{
  serial_cmd+=SERIAL_CMD_INCREMENT; //increment the serial command
  if(serial_cmd > MAX_SERIAL_CMD)
  {
    SERIAL_CMD_INCREMENT *= -1; //incrementally spin down the motors
  }

  if(serial_cmd <= 0)
  {
      // we are done!
      myQueue.scheduleRemoveFunction("ESC");
  }

  return serial_cmd;
}

void setup() {
    // Required for I/O from Serial monitor
    Serial.begin(9600);

    //ESC setup
    Serial.println("Initializing ESC");
    // Put the motors to Arduino pin 9 and 10
    myMotor.attach(9);
    myMotor2.attach(10);

    //IR Emitter Detector setup
    //Interrupt 0 is digital pin 2, so that is where the IR detector
is connected
    //Triggers on FALLING (change from HIGH to LOW)
    attachInterrupt(0, rpm_fun, FALLING);
    //Turn on IR LED
    pinMode(ledPin, OUTPUT);
    digitalWrite(ledPin, HIGH);
    //Use statusPin to flash along with interrupts
    pinMode(statusPin, OUTPUT);
    rpmcount = 0;
    rpm = 0;
    timeold = 0;
    status = LOW;

    //Function queue scheduling setup
    Serial.println("Initializing function queue scheduling");
    myQueue.scheduleFunction(esc, "ESC", 0, 3000);

   //Print table header
```

```
   Serial.println("Time(ms),RPM,Command");

   //Wait until start command i.e. any input serial comms
   Serial.println("Plug battery into motors, then send any serial
command");
   while(!Serial.available()) {}

    while(1) {
        myQueue.Run(millis());
        rpm_counter();
    }
}



//Receive ESC commands via serial
int esc(unsigned long now)
{
   int val = cmd(); //new rotation speed

   /*  We only want to write an integer between
    *  0 and 180 to the motor.
    */
   if (val > -1 && val < 181)
   {
       // Print confirmation that the
       // value is between 0 and 180
       // Write to Servo
       myMotor.write(val);
       myMotor2.write(val);
   }
}

//Count RPMs
void rpm_counter()
 {
   //Update RPM every second
   delay(1000);
   //Don't process interrupts during calculations
   detachInterrupt(0);
   //Note that this would be 60*1000/(millis() - timeold)*rpmcount if
the interrupt
   //happened once per revolution instead of twice. Other multiples
could be used
   //for multi-bladed propellers or fans
   rpm = 30*1000/(millis() - timeold)*rpmcount;
   timeold = millis();
   rpmcount = 0;

   //Write it out to serial port
   Serial.print(millis());
   Serial.print(",");
```

```
    Serial.print(rpm,DEC);
    Serial.print(",");
    Serial.println(serial_cmd);

    //Restart the interrupt processing
    attachInterrupt(0, rpm_fun, FALLING);
}

//not using the loop in this program
void loop() {
}
```

Results

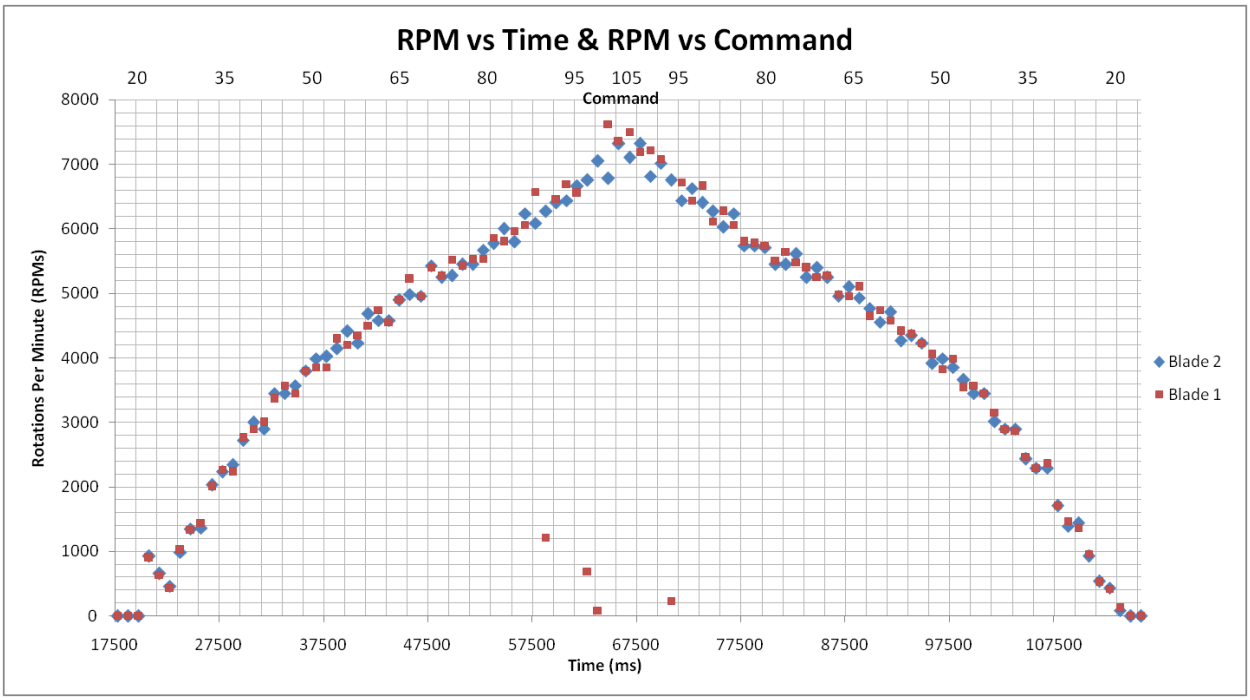
Blade 1

Time(ms)	RPM	Command
22408	0	25
23409	0	25
24410	0	25
25410	900	30
26411	638	30
27412	435	30
28412	1020	35
29414	1334	35
30414	1440	35
31415	2001	40
32416	2262	40
33417	2233	40
34418	2755	45
35419	2900	45
36419	3000	45
37421	3364	50
38421	3570	50
39422	3451	50
40423	3799	55
41424	3857	55
42425	3857	55
43425	4290	60
44427	4205	60
45427	4350	60
46429	4495	65
47429	4740	65
48430	4553	65
49431	4901	70
50432	5220	70
51433	4959	70
52433	5400	75
53435	5278	75
54435	5520	75
55436	5423	80
56437	5539	80
57438	5539	80
58439	5858	85
59440	5800	85
60440	5970	85
61442	6061	90
62442	6570	90
63443	1218	90
64444	6467	95
65445	6690	95
66446	6554	95
67446	690	100
68448	87	100
69448	7620	100
70450	7366	105
71450	7500	105
72451	7192	105
73452	7221	100
74452	7080	100
75454	232	100
76454	6720	95
77456	6438	95
78456	6660	95
79458	6119	90
80458	6270	90
81459	6061	90
82460	5800	85
83461	5771	85
84462	5742	85
85463	5510	80
86463	5640	80
87464	5481	80
88465	5394	75
89466	5249	75
90467	5278	75
91468	4988	70
92469	4959	70
93469	5100	70
94471	4640	65
95471	4740	65
96473	4582	65
97473	4410	60
98473	4380	60
99475	4234	60
100475	4050	55
101477	3828	55
102477	3990	55
103479	3538	50
104479	3570	50
105481	3451	50
106481	3150	45
107482	2900	45
108483	2871	45
109484	2465	40
110485	2291	40
111485	2370	40
112487	1711	35
113487	1470	35
114489	1363	35
115489	960	30
116491	522	30
117491	420	30
118492	116	25
119493	0	25
120494	0	25

Blade 2

Time(ms)	RPM	Command
17783	0	25
18784	0	25
19784	0	25
20786	928	30
21786	660	30
22787	464	30
23788	986	35
24788	1350	35
25789	1363	35
26790	2030	40
27791	2233	40
28792	2340	40
29793	2726	45
30793	3000	45
31795	2900	45
32795	3450	50
33797	3451	50
34797	3570	50
35799	3799	55
36799	3990	55
37799	4020	55
38801	4147	60
39801	4410	60
40803	4234	60
41803	4680	65
42804	4582	65
43805	4582	65
44806	4901	70
45807	4988	70
46808	4959	70
47808	5430	75
48809	5249	75
49810	5278	75
50811	5452	80
51812	5452	80
52812	5670	80
53814	5771	85
54814	6000	85
55816	5800	85
56816	6240	90
57818	6090	90
58818	6270	90
59819	6409	95
60820	6438	95
61820	6660	95
62822	6757	100
63822	7050	100
64824	6786	100
65824	7320	105
66826	7105	105
67826	7320	105
68828	6815	100
69828	7020	100
70829	6757	100
71830	6438	95
72830	6630	95
73832	6409	95
74832	6270	90
75834	6032	90
76834	6240	90
77835	5742	85
78836	5742	85
79837	5713	85
80838	5452	80
81839	5452	80
82839	5610	80
83841	5249	75
84841	5400	75
85842	5249	75
86843	4959	70
87843	5100	70
88845	4930	70
89845	4770	65
90847	4553	65
91847	4710	65
92849	4263	60
93849	4350	60
94850	4234	60
95851	3915	55
96851	3990	55
97853	3857	55
98853	3660	50
99854	3451	50
100855	3451	50
101856	3016	45
102857	2900	45
103858	2900	45
104859	2436	40
105860	2291	40
106861	2291	40
107862	1711	35
108863	1392	35
109863	1440	35
110865	928	30
111865	540	30
112866	435	30
113867	87	25
114868	0	25
115868	0	25

Plot



Video Presentation

http://www.youtube.com/watch?v=e57s49ww6dc

References

[1] http://www.instructables.com/id/Arduino-Based-Optical-Tachometer/

[2] http://techvalleyprojects.blogspot.com/2012/06/arduino-control-escmotor-tutorial.html

[3] https://github.com/Zuph/AVRQueue

[4] https://github.com/DuinOS/DuinOS