Internet of Things for What?

In my opinion the IoT is not so much about you controlling your toy helicopter with your smartphone. There's no fundamental difference with using a wireless controller. In my view it's more about machines informing and controlling each other without human intervention. Thinking of this I remembered the old model car track lying for decades in the attic at my parents. Apart from its fragility a major drawback was that I could never have more than one car on each track without them bumping in to each other at crossroads and junctions but also just because of speed differences running into each other's back. Also the little cars when run at a steady speed would either run too slow at the straight sections or too fast in the corners causing them to derail.

Objective

Connect an old Faller Ams modelcar track to a Windows Azure service via the internet using an Arduino computer. The Azure service contains an AI speed controller to adjust the running of the cars to prevent them from breaking out of corners and bumping into each other. Also it will adjust the speed for the cars to run faster at the straights and slower in the corners and put a maximum speed on specific cars so trucks will run slower than passenger cars. I've split this objective into 3 phases.

Phase I

Run a single car on a simple oval track. Control the speed and gather metric data while the car is running or standing still.

Phase II

Run two cars on the oval track divided into six separate sections. Control the speed of the cars not to bump into each other's backs based on the gathered metrics.

Phase III

Build a more complicated track with dozens of sections, crossroads and junctions and run half a dozen cars on it. Control the speed of the cars not only not to bump into each other's backs but also not to bump in to each other at the crossroads and junctions based on the gathered metrics.

The hardware

The Modelcar track

For those not acquainted with the Faller Ams model car track visit this website:
http://www.everyoneweb.com/falleramsautos

The Arduino Starter kit

I've chosen for an Arduino minicomputer as it's very basic and provides a lot of different connection possibilities and also it's easily extendable with additional components like the Ethernet shield I'm using to connect it to the internet. As I'm not proficient with electronics I've bought a starters kit that ships with a lot of basic components and simple examples.

Windows Azure and Data*Spider*

For the messaging system scaffolding I use my self-developed DataSpider framework. The service runs in Windows Azure. For logging and collecting the metric data there are a couple of SqlServer Azure databases.

Windows Azure

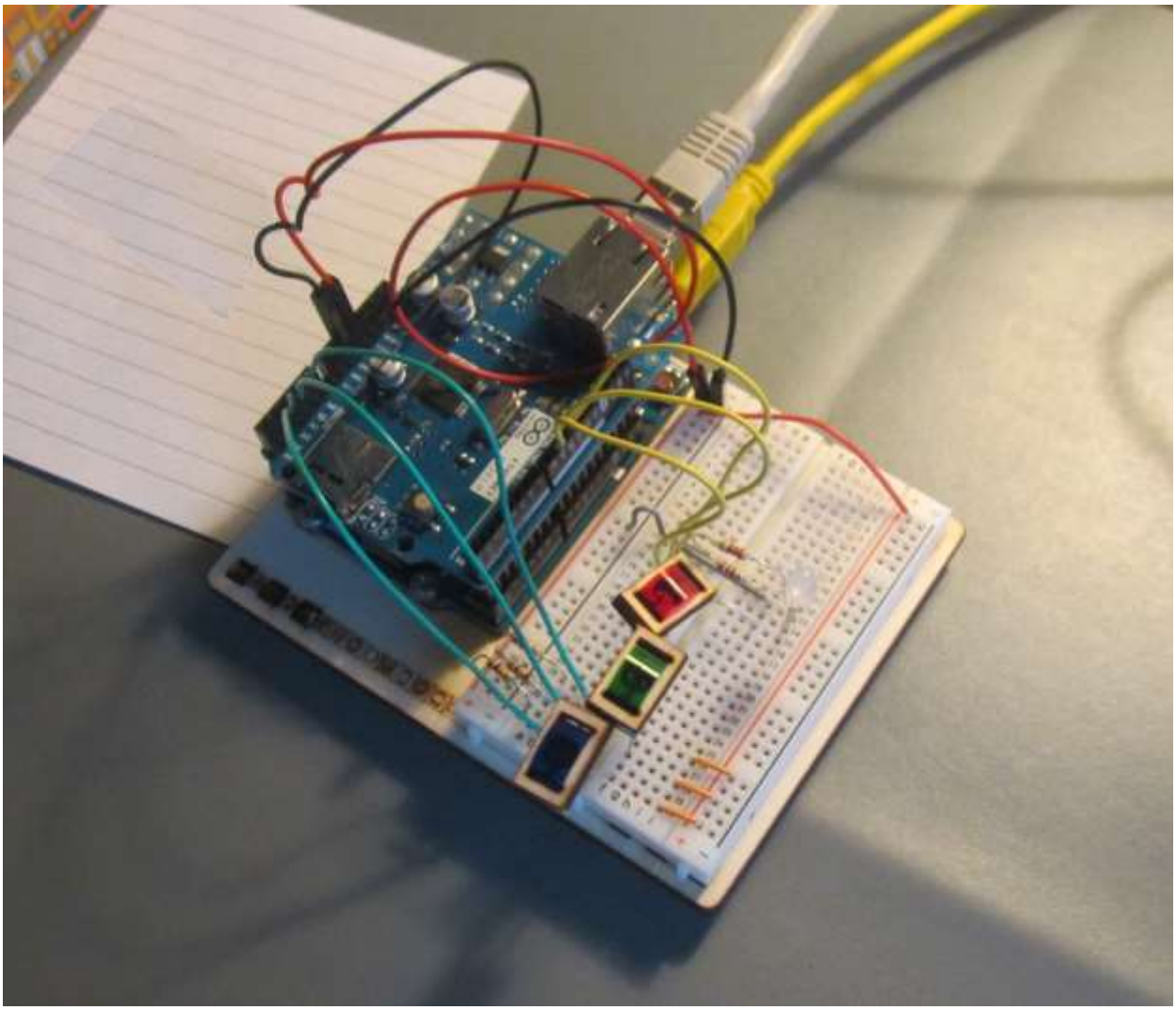
DataSpider
Connects software to software

Progress and issues

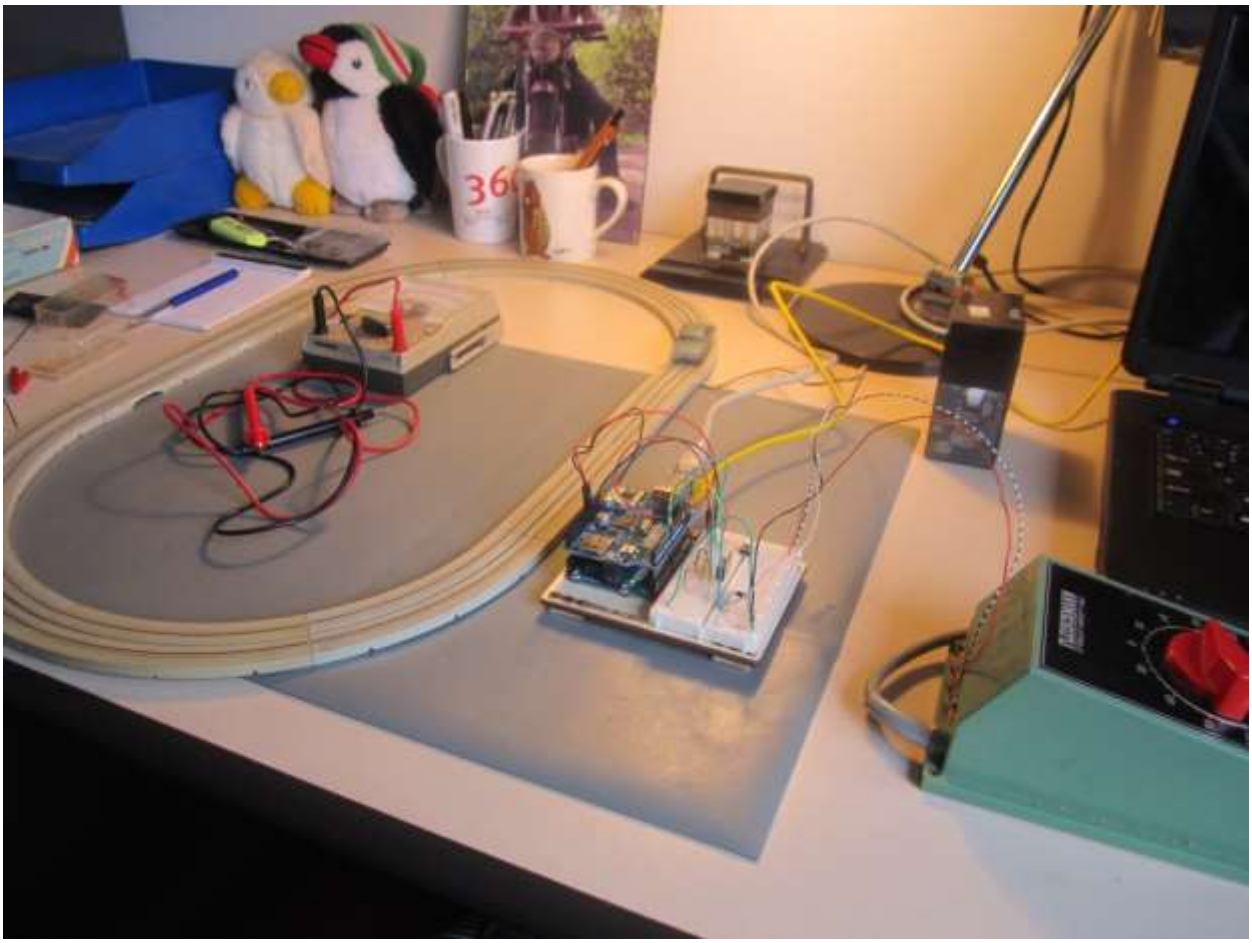
December 2014 after collecting the car track from the attic at my parents it needed cleaning and some repairs on the junctions electrics. It took a while to get the cars running again more or less. I had to do cleaning and oiling of the electric motors and some soldering of broken or disconnected parts.

For getting acquainted with the Arduino I build some of the example projects. This is a lot of fun and gives you a quick start on how it works.

Connect the Arduino to the network. For this you need an Ethernet shield that sits on top of the Arduino:



Januari 2015 connect the car track to the Arduino: The Arduino runs at 5 volts and the car track needs around 10 volts to run smoothly. For getting around this is the mosfet that can be regulated with 5 volts and can withstand the 10 volts from the power supply to run through it. To gather current data from the track I use a Hall effect sensor for the same reason as it can withstand the 10 volts/200 mAmp and provides a 2.5 volts signal to the Arduino. As already stated I'm not proficient with this stuff so it took a lot of internet search to piece this together.

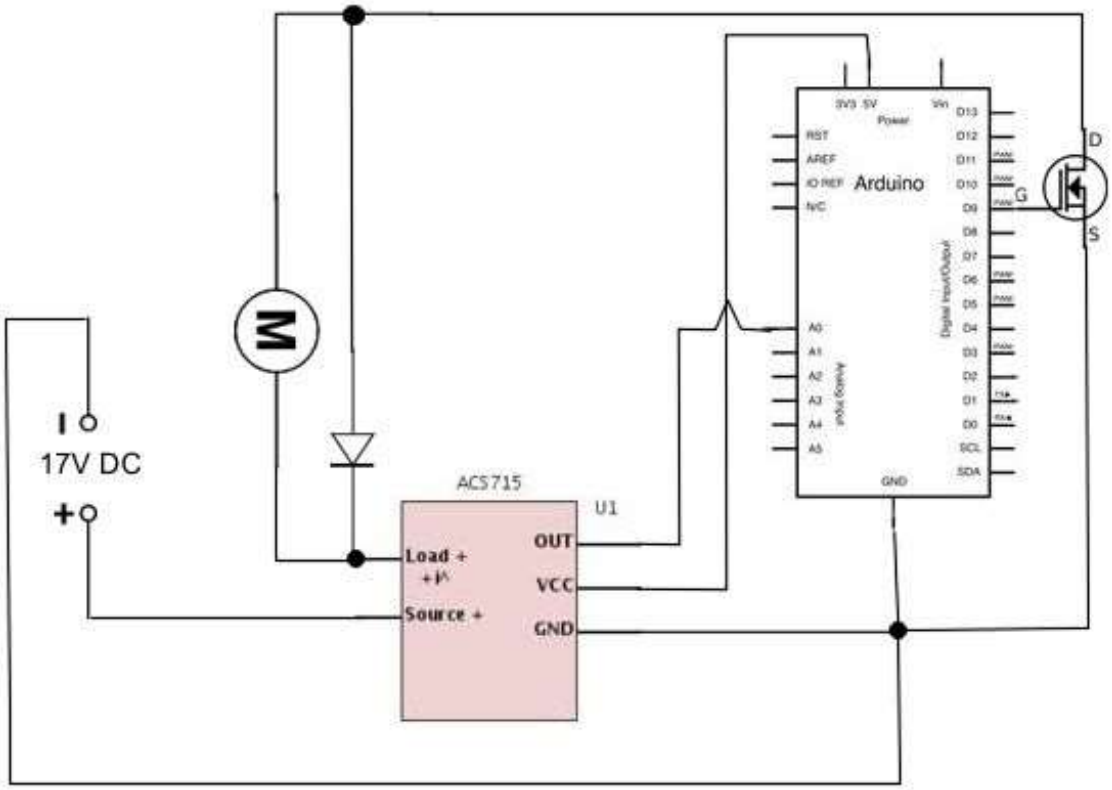


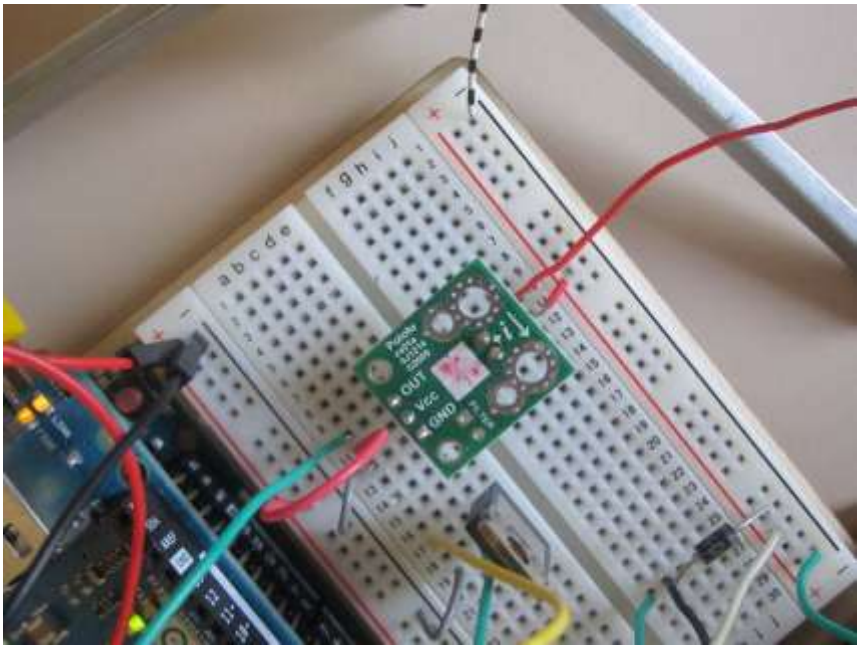
With the Fleischmann DC power supply hooked up and the digital PWM output attached to the mosfet transistor allows the speed to be controlled from a little command-line program running on the PC. A Hall effect sensor (Pololu **ACS715**) allows for reading the current, in the chart below you can see the current runs on a baseline until a car is running on the track when it goes up. Issues encountered are mostly cars stalling due to electrics connection problems, uneven track, dirt etc. The Hall effect sensor consequently gives uneven output making calculations unreliable.

Basic Setup

Faller Ams Track

The Faller Ams track is connected to the Arduino via the electric components. Basic power supply is from an old Fleischmann model train track supplying between 0-17 volts





The Hall effect sensor.

Arduino embedded C/C++ code

The Arduino is very limited in resources and is programmed in C/C++. Also my C/C++ is not really that great so I needed to keep processing at the Arduino to an absolute minimum and get the data to the Windows Azure service as simple and fast as possible.

```
#include <SPI.h>
#include <Ethernet.h>

// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network, and it's optional if DHCP is enabled
byte mac[] = { 0x90, 0xA2, 0xDA, 0x0D, 0xBC, 0xAE };
//IPAddress ip(192,168,15,177); //We're not going to use this, it's just for reference
static const byte deviceId[] = { 0x00, 0x00, 0x00, 0x01 };
EthernetClient client;
// named constants for the switch and motor pins
const int motorPin =  9; // the number of the motor pin
const int sensorPin =  A0; // the number of the motor pin
float val = 0;
float iloop = 0;
byte bytes = 196;
//The union is for converting the float to a byte array
typedef union {
  float floatingPoint;
  byte binary[4];
} binaryFloat;
binaryFloat v;
//Serial.print statements are for debugging on the usb connection with the monitor on the notebook

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
  Ethernet.begin(mac);
  //Ethernet.begin(mac, ip);
}

void loop(){
    if (! client.connected()){
        Serial.println("Trying to connect");
        //char* host = "cloudspider.cloudapp.net";
        char* host = "192.168.178.28";
        client.setTimeout(1000);
        client.connect(host, 10100);
        if (client.connected()) {
            Serial.println("Connected to port, writing deviceId and waiting for commands...");
            client.write(deviceId, sizeof(deviceId));
        }
    }
    // write the bytes returned from the remote service on the motorpin
    // this will regulate the speed of the car
    analogWrite(motorPin, bytes);
    // read the analog in value from the sensor to send to the remote service
    // so it will know if the car is actually running
    if(iloop < 100){
      val = val + analogRead(sensorPin);
      iloop++;
      delay(10);
    }
    else{
      Serial.print("bytes = " );
      Serial.print(bytes);
      Serial.print("\t amps = ");
      Serial.print(val/iloop-510);
```

```
      Serial.print("\t rest = ");
      Serial.print((val/iloop)/(bytes/100));
      //we have very limited memory so keep track of what we're using
      Serial.print("\t ram = ");
      Serial.println(freeRam());

      if(client.connected()){
        // average a number of readings to flatten out the value a bit
        v.floatingPoint = val/iloop-510;
        Serial.print(v.binary[0]);
        Serial.print("\t");
        Serial.print(v.binary[1]);
        Serial.print("\t");
        Serial.print(v.binary[2]);
        Serial.print("\t");
        Serial.println(v.binary[3]);
        //send the value to the remote service
        client.write(v.binary, sizeof(v.binary));

        byte buff[1];
        //read the reply back in
        if(client.read(buff, sizeof(buff)) > 0){
          bytes = buff[0];
        }
      }
      val = 0;
      iloop = 0;
    }
}

int freeRam () {
  extern int __heap_start, *__brkval;
  int v;
  return (int) &v - (__brkval == 0 ? (int) &__heap_start : (int) __brkval);
}
```

Windows Azure service C# code

The TCP connection to pick up the Arduino messages:

```
public override void Listen() {
                        _listener = new TcpListener(new IPEndPoint(IPAddress.Any,_cconfig.Port));
                        _listener.Start();
                        listenForClients(_listener);

                }

                private async void listenForClients(TcpListener tcpServer)
                {
                        while (_running)
                        {
                                TcpClient tcpClient = await tcpServer.AcceptTcpClientAsync();
                                base.On_Signal(new DSSignalEventArgs(this.ToString(),"Message
Received",(int)eLogLevel.Informational));
                                processClient(tcpClient);
                        }
                }

                private async void processClient(TcpClient tcpClient)
                {
                        NetworkStream stream = tcpClient.GetStream();
                        byte[] buffer = new byte[_cconfig.BufferSize];
                        int read = await stream.ReadAsync(buffer, 0, _cconfig.BufferSize);
```

```
                        _tmpmessage = null;
                        _waiting = true;
                        DSMessage message = DSMessageHooks.InitializeDSMessage();
                        message.DSBody.Message.Body.Data = buffer;
        message.DSBody.Message.Header.ProcessorID = _cconfig.ProcessorId;
                        base.On_Receipt(new ReceiptEventArgs(message));

                        while(_waiting && _running){} //wait for the return value will not take long
                        message = _tmpmessage;

                        await
stream.WriteAsync((byte[])message.DSBody.Message.Body.Data,0,((byte[])message.DSBody.Message.Body.Data).Le
ngth);
                        base.On_Signal(new DSSignalEventArgs(this.ToString(),"Message
Send",(int)eLogLevel.Informational));
                }
```

For further analysis I want the data in a SqlAzure database so I can run queries and reports on it. Of course the Sql Azure code is just plain SqlServer connection code:

```
public override object SendReceive(object OutgoingMessage) {
                        int reply = 0;
                        DSMessage message = (DSMessage)OutgoingMessage;

                        using(SqlConnection conn = new SqlConnection(_config.ConnectString)){
                                conn.Open();
                                SqlCommand command = conn.CreateCommand();
                                command.CommandType = System.Data.CommandType.Text;
                                command.CommandText = (string)message.DSBody.Message.Body.Data;
                                SqlDataReader reader = command.ExecuteReader();
                                if(reader.Read()){
                                        reply = reader.GetInt32(0);
                                }
                                conn.Close();
                        }
                        message.DSBody.Message.Body.Data = reply;
                        return message;
                }
```

The Windows Azure service currently doesn't do much besides storing the data in the SqlServer databases. This logic will be expanded in the next project phase.

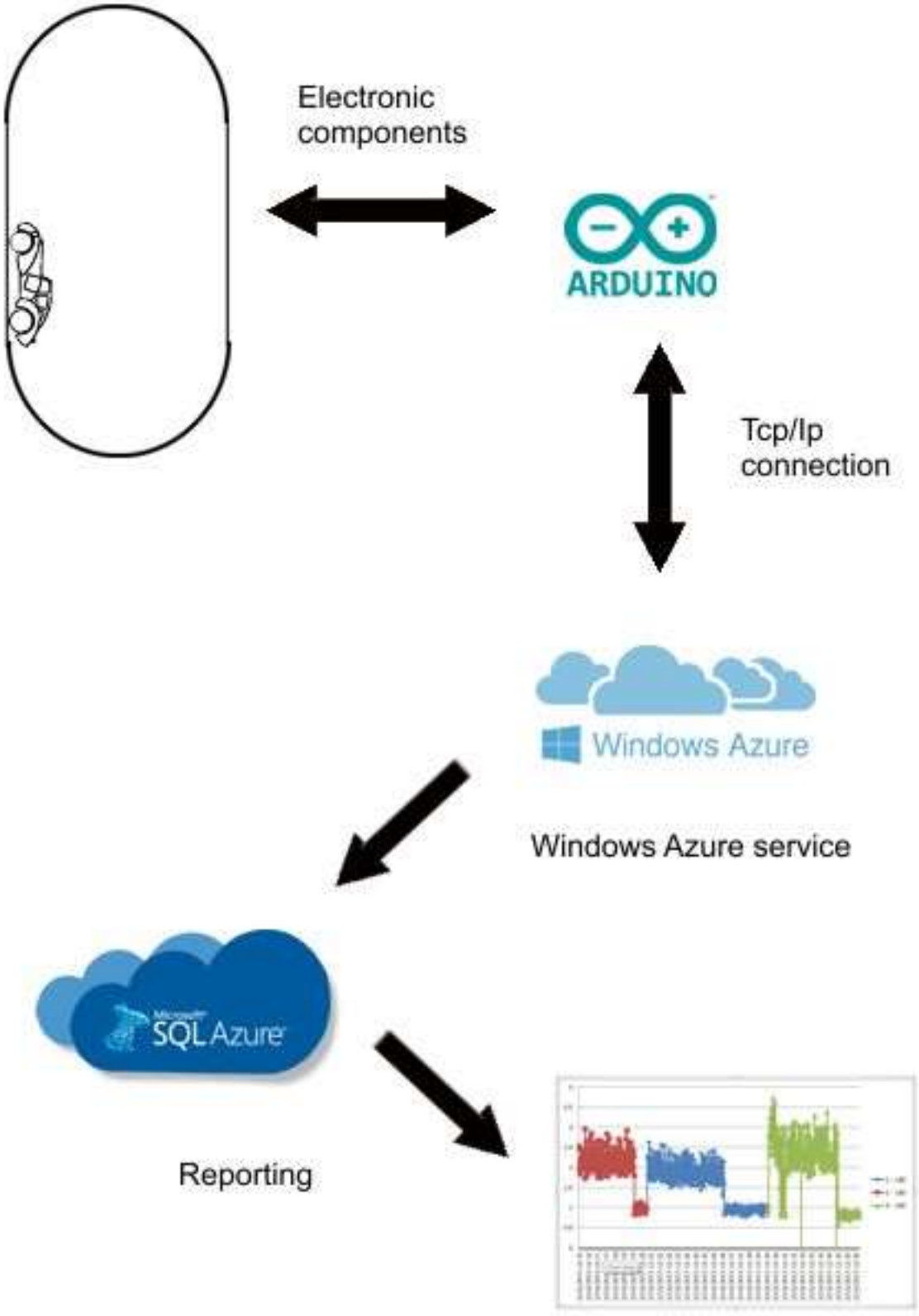
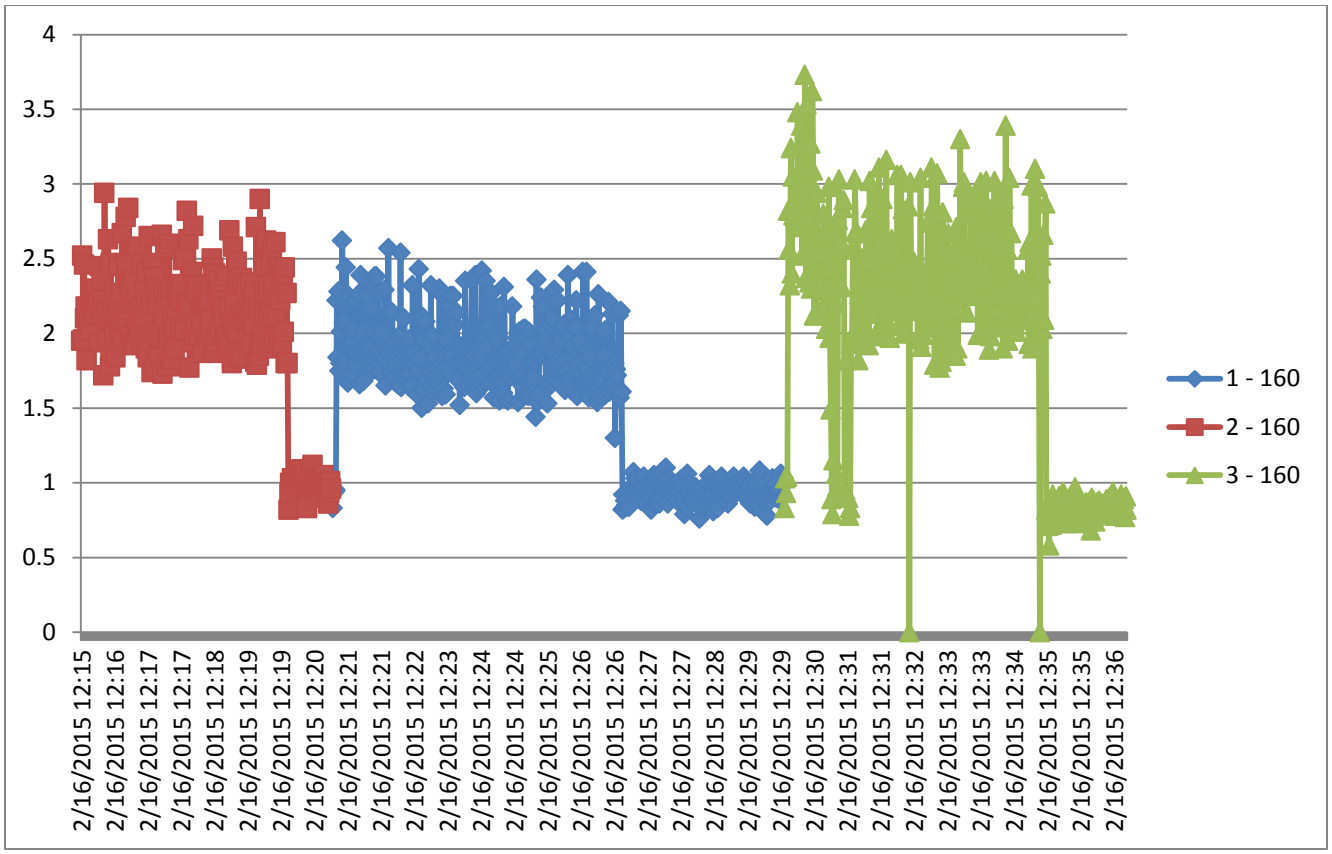


Diagram of the current setup.



(The numbers in the graph don't represent 'real' voltage or current values.)

In the graph three different cars have been measured while running with short idle gaps in between. As you can easily see from the graph, the gathered values differ in a rather broad range and as a consequence the cars cannot reliably be differentiated from each other. The gap with idle condition however is big enough to determine if anything is running or not. The couple of low readings on the 'green' car are due to stalling and derailing.

With this setup so far the first phase is done.