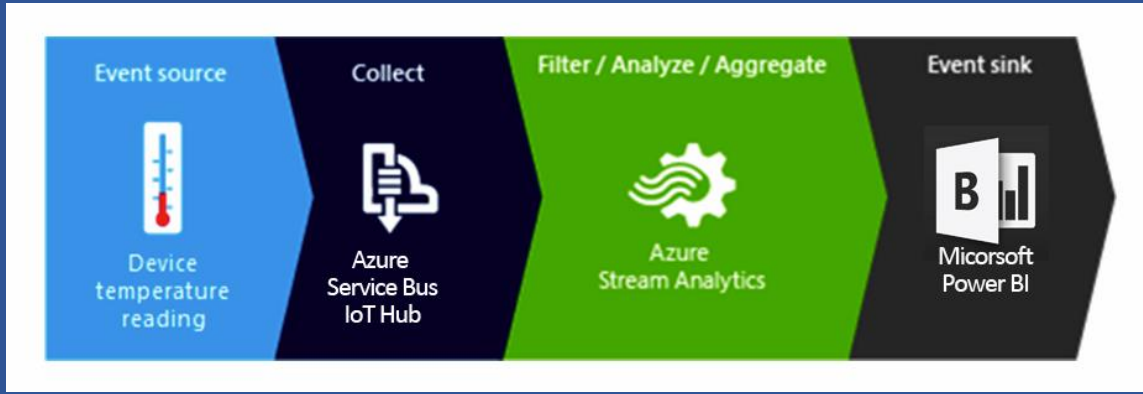
Create Motor Sim

Working steps

1. Create Motor Sim
2. Sending telemetry data from Motor device Sim to Azure IoT Hub
3. Send data to Azure Stream Analytics
4. Sink event to BI
5. Create data visualization in Microsoft Power BI



Question

Is the motor run normally?

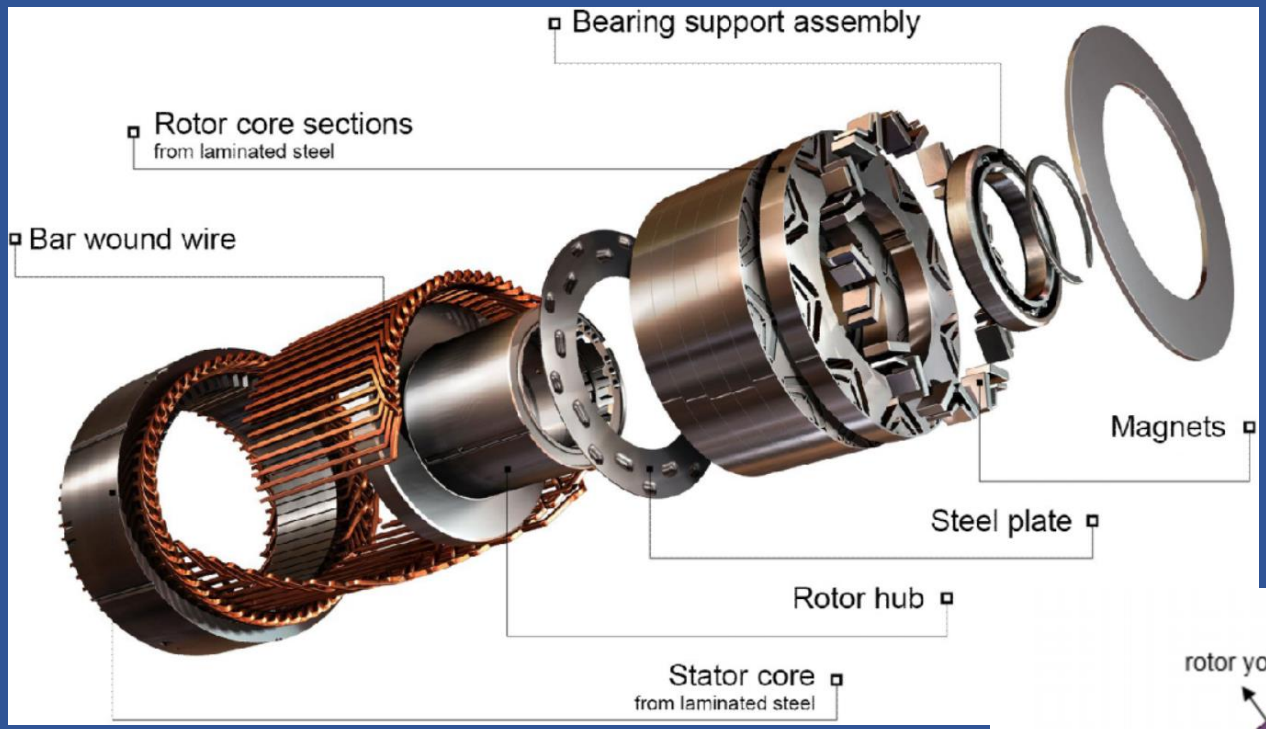
Is the motor speed right?

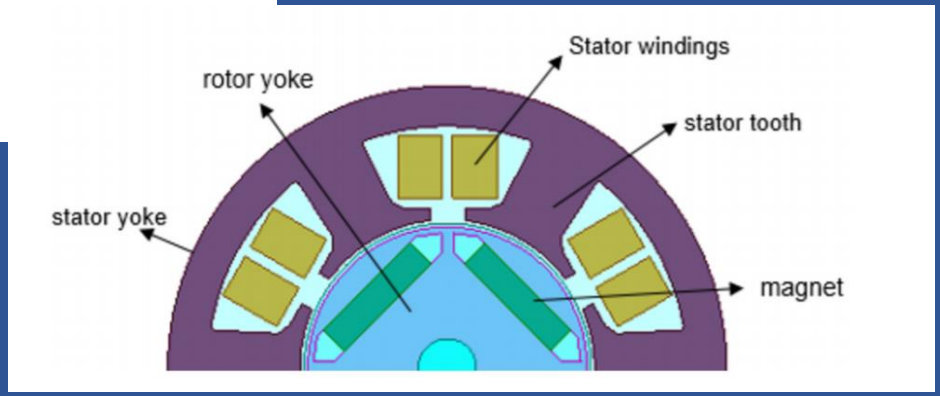
Is the motor need maintenance?

Application

Preventive maintenance

Motor's anatomy





Context

- The dataset comprises several sensor data collected from a permanent magnet synchronous motor (PMSM) deployed on a test bench.
- The PMSM represents a german OEM's prototype model.
- Test bench measurements were collected by the LEA department at Paderborn University. This dataset is mildly anonymized.

Content

- All recordings are sampled at 2 Hz.
- The dataset consists of multiple measurement sessions, which can be distinguished from each other by column "profile_id".
- A measurement session can be between one and six hours long.
- The motor is excited by hand-designed driving cycles denoting a reference motor speed and a reference torque.
- Currents in d/q-coordinates (columns "i_d" and i_q") and voltages in d/q-coordinates (columns "u_d" and "u_q") are a result of a

standard control strategy trying to follow the reference speed and torque.

- Columns "motor_speed" and "torque" are the resulting quantities achieved by that strategy, derived from set currents and voltages.
- Most driving cycles denote random walks in the speed-torque-plane in order to imitate real world driving cycles to a more accurate degree than constant excitations and ramp-ups and -downs would.

Motor's sensors data

ambient	coolant	u_d	u_q	motor_speed	torque	i_d	i_q	pm	stator_yoke	stator_tooth	stator_winding
-0.75214	-1.11845	0.32794	-1.29786	-1.2224282	-0.25018	1.02957	-0.24586	-2.52207	-1.8314217	-2.0661428	-2.0180326
-0.77126	-1.11702	0.32966	-1.29769	-1.2224293	-0.24913	1.02951	-0.24583	-2.52242	-1.8309687	-2.0648587	-2.0176313
-0.78289	-1.11668	0.33277	-1.30182	-1.2224278	-0.24943	1.02945	-0.24582	-2.52267	-1.8304	-2.064073	-2.0173435
-0.78094	-1.11676	0.3337	-1.30185	-1.2224301	-0.24864	1.03284	-0.24695	-2.52164	-1.8303328	-2.0631368	-2.0176322
-0.77404	-1.11678	0.33521	-1.30312	-1.2224286	-0.2487	1.03181	-0.24661	-2.5219	-1.8304977	-2.0627947	-2.0181448
-0.76294	-1.11695	0.3349	-1.30302	-1.2224286	-0.2482	1.03103	-0.24634	-2.5222	-1.8319309	-2.0625494	-2.017884
-0.74923	-1.11617	0.33501	-1.30208	-1.2224296	-0.24791	1.03049	-0.24616	-2.52254	-1.8330117	-2.0621152	-2.0172427
-0.73845	-1.11399	0.33626	-1.30515	-1.2224321	-0.24832	1.03011	-0.24603	-2.52284	-1.8321822	-2.0619526	-2.0172133
-0.73091	-1.11183	0.33491	-1.30379	-1.2224315	-0.24778	1.02985	-0.24598	-2.52281	-1.8315759	-2.062443	-2.0177386
-0.72713	-1.10949	0.33599	-1.30563	-1.2224314	-0.24829	1.02964	-0.24589	-2.52268	-1.8314383	-2.062317	-2.0181801
-0.72371	-1.10828	0.3354	-1.30456	-1.2224283	-0.24791	1.02951	-0.24583	-2.52263	-1.8314928	-2.0620575	-2.0176919
-0.71775	-1.10859	0.33443	-1.30434	-1.2224288	-0.24772	1.02939	-0.2458	-2.52264	-1.8318189	-2.0622487	-2.017435

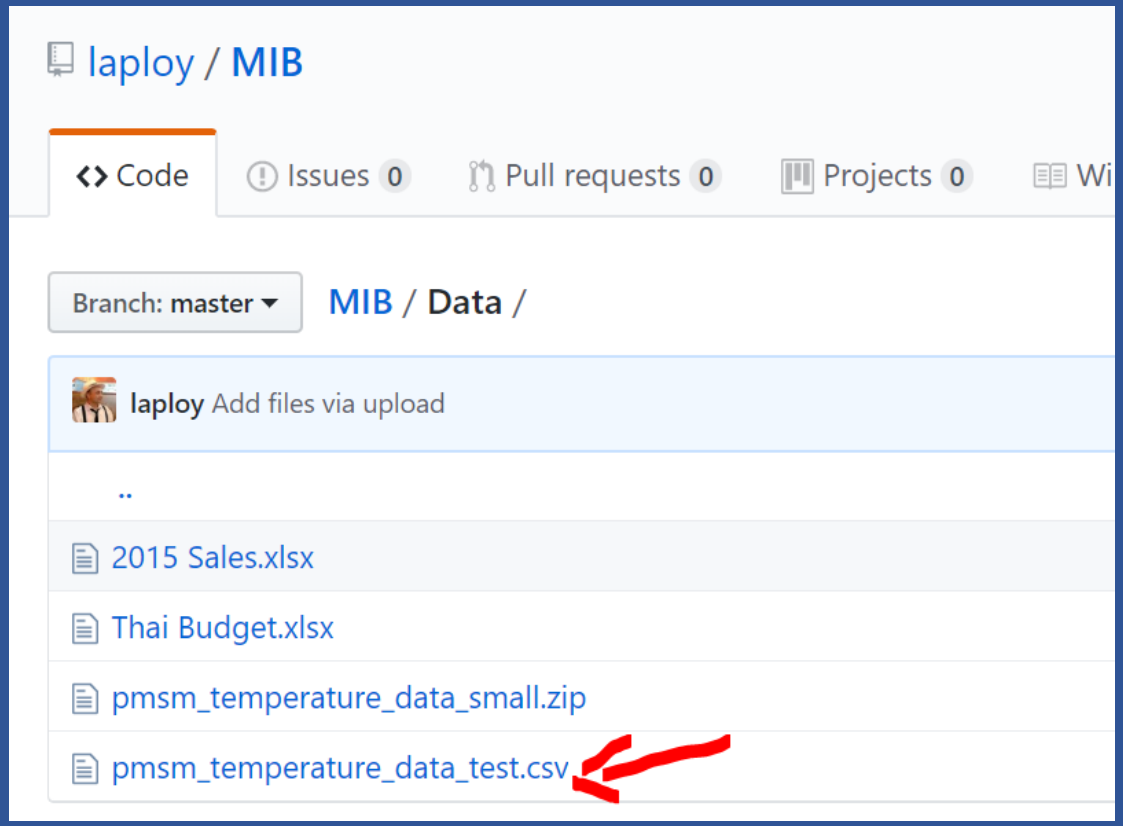
Motor's parameters

- ambient: Ambient temperature as measured by a thermal sensor located closely to the stator.
- coolant: Coolant temperature. The motor is water cooled. Measurement is taken at outflow.
- u_d: Voltage d-component
- u_q: Voltage q-component
- motor_speed: Motor speed
- torque: Torque induced by current.
- i_d: Current d-component

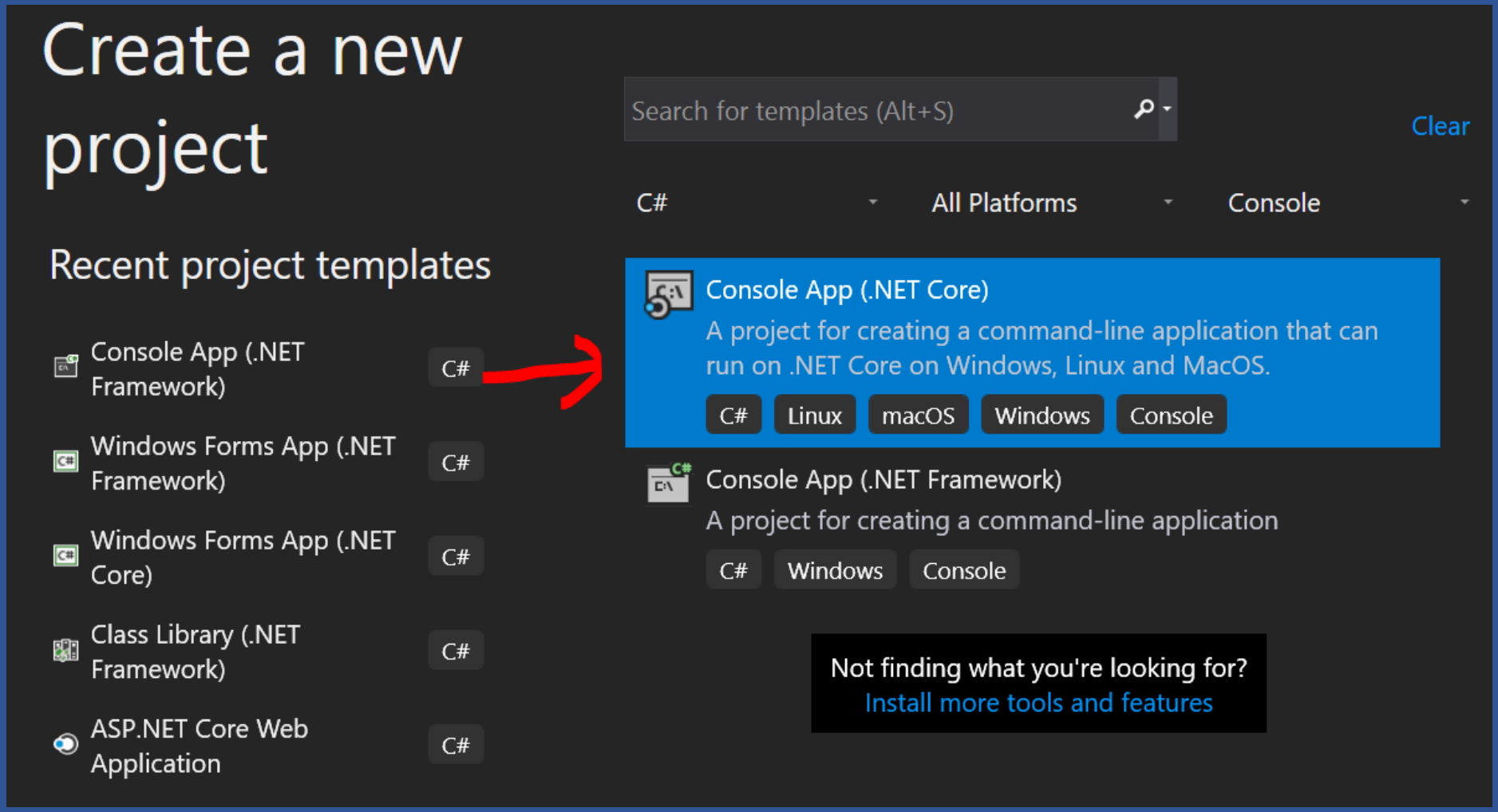
- i_q: Current q-component
- pm: Permanent Magnet surface temperature representing the rotor temperature. This was measured with an infrared
- stator_yoke: Stator yoke temperature measured with a thermal sensor.
- stator_tooth: Stator tooth tempera7ture measured with a thermal sensor.
- stator_winding: Stator winding temperature measured with a thermal sensor.
- profile_id: Each measurement session has a unique ID. Make sure not to try to estimate from one session onto the other as they are

Get test dataset

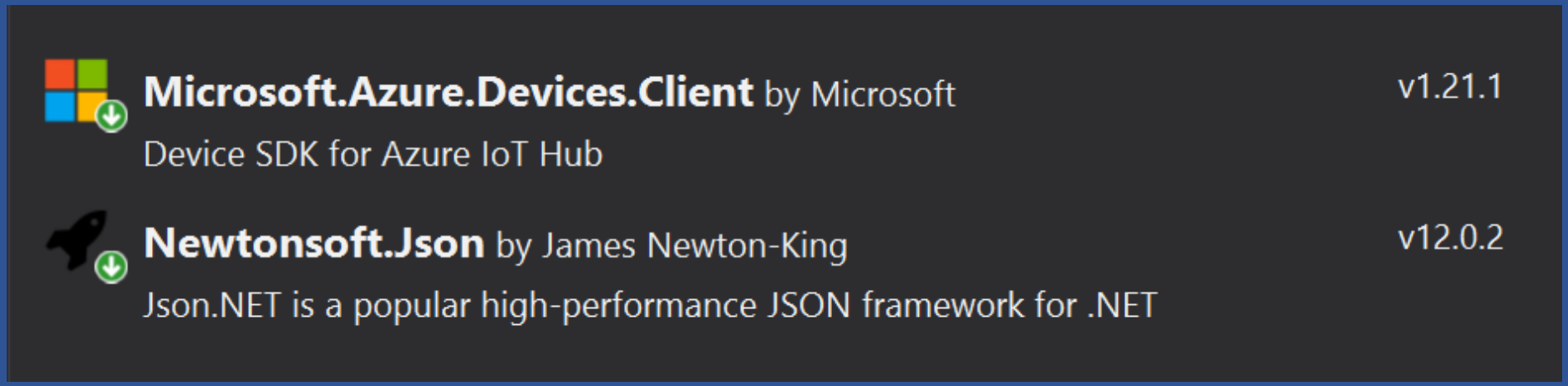
- Create folder D:\MIB
- Download file pmsm_temperature_data_test.csv
- Save in D:\MIB



Open Visual Studio / Create C# Console App .NET Core / Name = motor



NuGet two packages



Add class Motor

```
 99    public class Motor
100    {
101        public float ambient { get; set; }
102        public float coolant { get; set; }
103        public float u_d { get; set; }
104        public float u_q { get; set; }
105        public float motor_speed { get; set; }
106        public float torque { get; set; }
107        public float i_d { get; set; }
108        public float i_q { get; set; }
109        public float pm { get; set; }
110        public float stator_yoke { get; set; }
111        public float stator_tooth { get; set; }
112        public float stator_winding { get; set; }
113    }
114 }
```

Add namespace

```
6   using Microsoft.Azure.Devices.Client;
7   using Newtonsoft.Json;
8   using System;
9   using System.Text;
10  using System.IO;
11  using System.Collections.Generic;
```

Add fields

```
17      private static List<string[]> dataset;
18      private static int counter = 0;
19      private static DeviceClient myDevice;
20      private readonly static string connectionString =
21          "HostName=loyiothub1.azure-devices.net;DeviceId=loy-iot-
```

Add method ReadCSV()

```
37      private static void ReadCSV()
38      {
39          using (var reader = new StreamReader(
40              @"g:\temp\pmsm_temperature_data_test.csv"))
41          {
42              dataset = new List<string[]>();
43              while (!reader.EndOfStream)
44              {
45                  var line = reader.ReadLine();
46                  var values = line.Split(',');
47                  dataset.Add(values);
48              }
49          }
50      }
```

Add method GetData()

```
51      private static Motor GetData()
52      {
53          if (counter++ > 100) counter = 0;    // dataset is 100
54          var v = dataset[counter];   // get array of one telem
55          if (v.Length < 12) return null;  // telemerty must ha
56          Motor myMotor = new Motor();
57          int i = 0;  // array index
58          myMotor.ambient = float.Parse(v[i++]);
59          myMotor.coolant = float.Parse(v[i++]);
60          myMotor.u_d = float.Parse(v[i++]);
61          myMotor.u_q = float.Parse(v[i++]);
62          myMotor.motor_speed = float.Parse(v[i++]);
63          myMotor.torque = float.Parse(v[i++]);
64          myMotor.i_d = float.Parse(v[i++]);
65          myMotor.i_q = float.Parse(v[i++]);
66          myMotor.pm = float.Parse(v[i++]);
67          myMotor.stator_yoke = float.Parse(v[i++]);
68          myMotor.stator_tooth = float.Parse(v[i++]);
69          myMotor.stator_winding = float.Parse(v[i++]);
70          return myMotor;
71      }
```

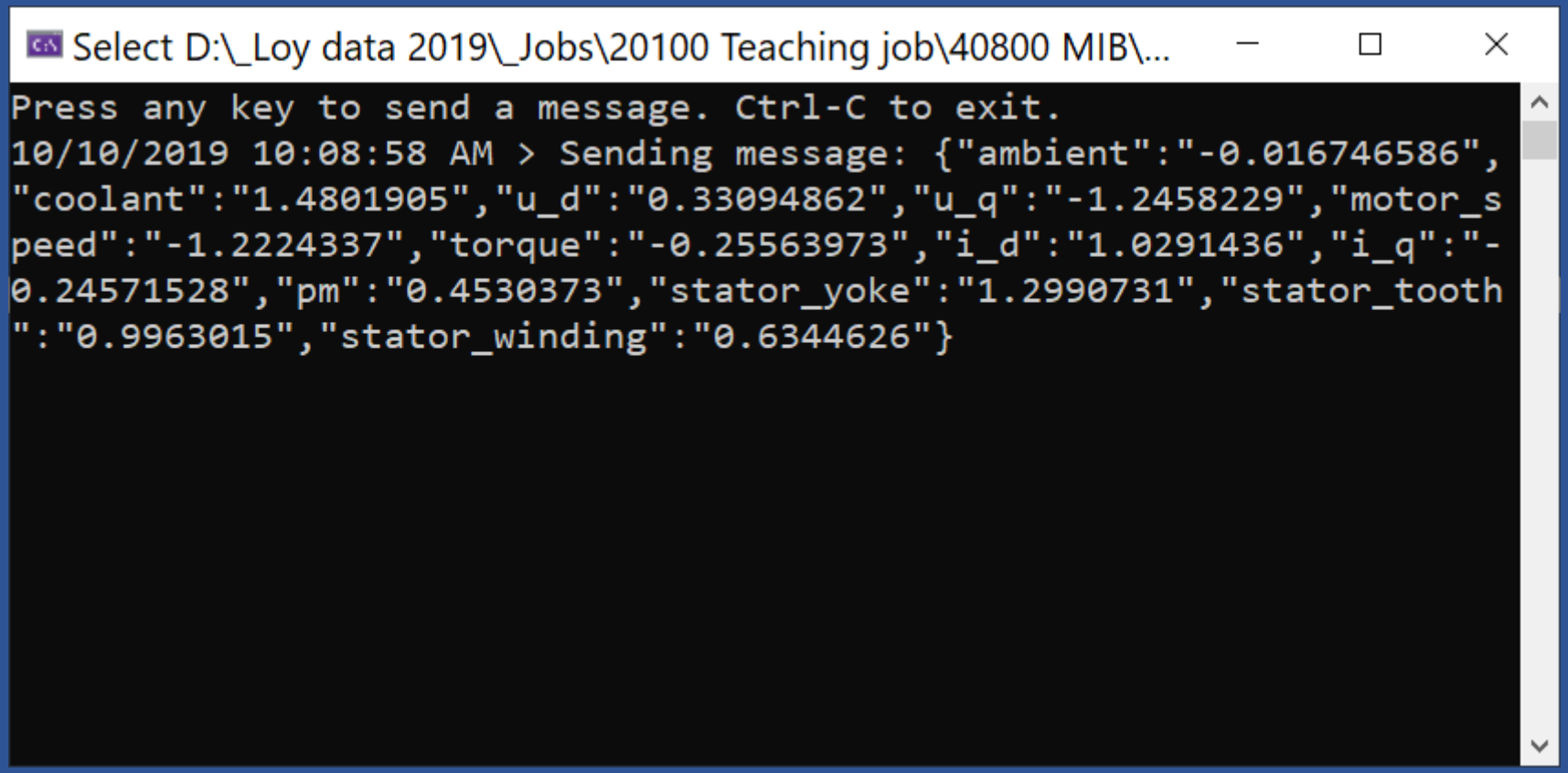
Add method SendD2C()

```
72          private static async void SendD2C()
73          {
74              var motor = GetData();
75              if (motor == null) return;
76              var telemetry = new
77              {
78                  ambient = motor.ambient,
79                  coolant = motor.coolant,
80                  u_d = motor.u_d,
81                  u_q = motor.u_q,
82                  motor_speed = motor.motor_speed,
83                  torque = motor.torque,
84                  i_d = motor.i_d,
85                  i_q = motor.i_q,
86                  pm = motor.pm,
87                  stator_yoke = motor.stator_yoke,
88                  stator_tooth = motor.stator_tooth,
89                  stator_winding = motor.stator_winding
90              };
91              var messageString = JsonConvert.SerializeObject(telemetry);
92              var message = new Message(Encoding.ASCII.GetBytes(messageString));
93              await myDevice.SendEventAsync(message); // Send the telemetry message
94              Console.WriteLine("{0} > Sending message: {1}", DateTime.Now, messageString);
95          }
```

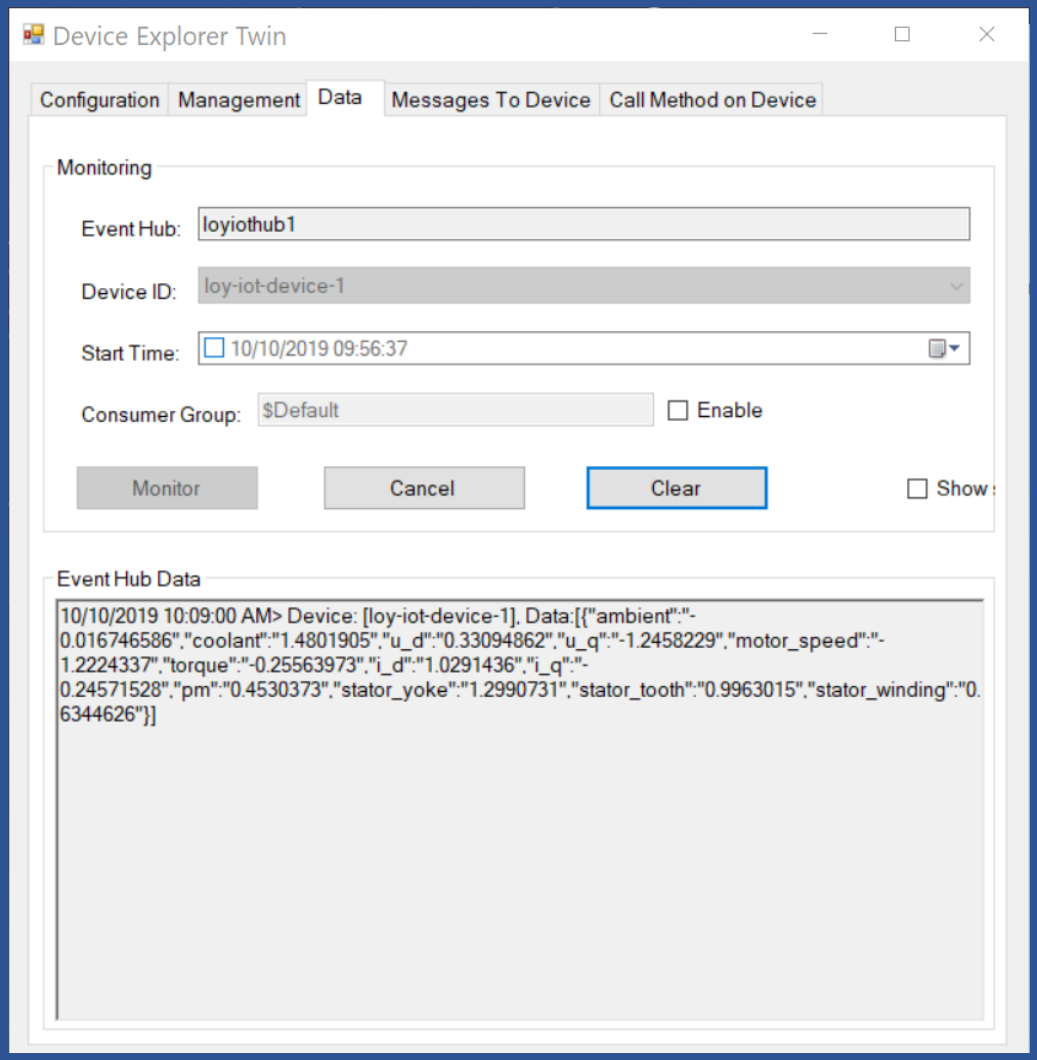
Add code to Main

```
23      private static void Main(string[] args)
24      {
25          ReadCSV();  // read test dataset
26          // Connect to the IoT hub using the MQTT protocol
27          myDevice = DeviceClient.CreateFromConnectionString(
28              connectionString,
29              TransportType.Mqtt);
30          while (true)
31          {
32              Console.WriteLine("Press any key to send a message. Ctrl-C to exit.");
33              Console.ReadLine();
34              SendD2C();
35          }
36      }
```

Run and verify



Monitor data in Device Explorer



Make sure data is right

10/10/2019 10:09:00 AM>
Device: [loy-iot-device-1], Data:[{
"ambient":"-0.016746586",
"coolant":"1.4801905",
"u_d":"0.33094862",
"u_q":"-1.2458229",
"motor_speed":"-1.2224337",
"torque":"-0.25563973",
"i_d":"1.0291436",
"i_q":"-0.24571528",
"pm":"0.4530373",
"stator_yoke":"1.2990731",
"stator_tooth":"0.9963015",
"stator_winding":"0.6344626"}]

What's next?