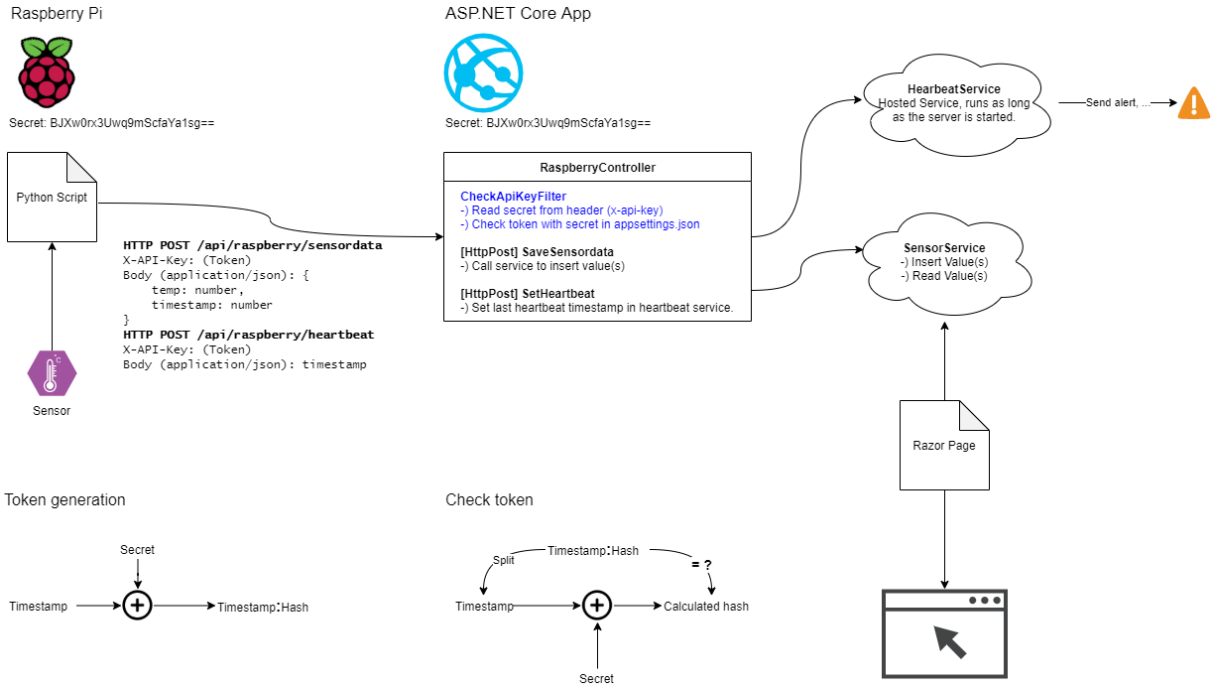
Raspberry PI und ASP.NET Core API Demonstration

Starten des Python Clients und des Webservers

Zur Demonstration muss auf einem PC Python 3 und .NET Core 5 installiert werden. Client und Server werden mit folgenden Befehlen gestartet:

```
cd SensorDemo.Raspberry
start python main.py
cd ..\SensorDemo.Webapp
dotnet run
```

Schema der Applikation



Sicherheit

Es wird im HTTP Header *X-API-Key* der Zeitstempel samt Hash gesendet. Dies sieht z. B. so aus: *1022324234:dsf349fjsdfkg*. Der Hash wird mit dem konfigurierten Secret erzeugt. Dadurch kann der Token von keinem unberechtigten erzeugt oder verändert werden, da der Hash dann nicht stimmen würde. Der Token hat eine definierte Gültigkeit, die in der Klasse CheckApiKeyFilterAttribute in der Rest API eingestellt wird. Achtung: Laufen die Uhren von Raspberry und Server über dieses Zeitmaß auseinander, wird kein Request mehr funktionieren.

Der *Heartbeat* wird im Background Service HearbeatService geprüft. Es wird mit dem Server gestartet. Der Controller setzt im Service den letzten empfangenen Zeitstempel. Ist dieser zu alt, kann das Service entsprechende Aktionen auslösen.

Das Hearbeat Service

Da das HearbeatService über DI vom Controller verwendet wird (setzen des letzten Beats), ist es kein klassisches Hosted Service. Es ist ein Singleton, welches in der Main Methode in Program.cs zu Beginn gestartet wird:

```
public static void Main(string[] args)
{
    var host = CreateHostBuilder(args).Build();
    // using Microsoft.Extensions.DependencyInjection;
    var service = host.Services.GetRequiredService<HearbeatService>();
```

```
    service.StartHearbeatService();
    host.Run();
}
```

Danach wird es in *ConfigureServices* normal als Singleton registriert:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<HearbeatService>();
    services.AddTransient(provider => new Controllers.CheckApiKeyFilterAttribute(
        secret: Configuration["Secret"]));
    // ...
}
```

Der Filter CheckApiKeyFilter

Der Controller RaspberryController wird durch einen Filter geschützt. Er ist in der Klasse CheckApiKeyFilterAttribute definiert und ist ein Service Filter, da er das Secret über DI bekommen muss. Deswegen wurde er auch mit *AddTransient* in *ConfigureServices* registriert.