Projet N2 (Web Phidgets)

Table des matières

[Project description 2](#_Toc433702412)

[Exploitation 4](#_Toc433702413)

[Architecture 5](#_Toc433702414)

[Visual studio 2015 solution organization 6](#_Toc433702415)

[Ja.Mvc.SignalR.Phidget solution projects. 6](#_Toc433702416)

[Ja.Mvc.SignalR.Phidget 6](#_Toc433702417)

[Ja.SignalRConsole.PhidgetServer 7](#_Toc433702418)

[Ja.SignalR.PhidgetServer 7](#_Toc433702419)

[Ja.SignalR.Phidget.Models 7](#_Toc433702420)

[Code 8](#_Toc433702421)

[Considération générales relatives au fonctionnement 8](#_Toc433702422)

[Ja.SignalR.Phidget.Models 10](#_Toc433702423)

[DTO Phidget Module 11](#_Toc433702424)

[DTO SignalRClientDetails 16](#_Toc433702425)

[Ja.Mvc.SignalR.Phidget 17](#_Toc433702426)

[SignalR server 17](#_Toc433702427)

[SignalR client 19](#_Toc433702428)

[Ja.SignalRConsole.PhidgetServer 20](#_Toc433702429)

[Ja.SignalR.PhidgetServer 21](#_Toc433702430)

[Glossaire: 23](#_Toc433702431)

# Project description

Le but du projet est de permettre aux utilisateurs d’un client léger (navigateur Internet) [PhidgetClient], de gérer à distance un nombre quelconque de modules Phidget <http://www.phidgets.com/> en temps réel.

Un module Phidget par machine (desktop, embarqué…) étant attaché à un port USB de cette dernière [PhidgetServer].

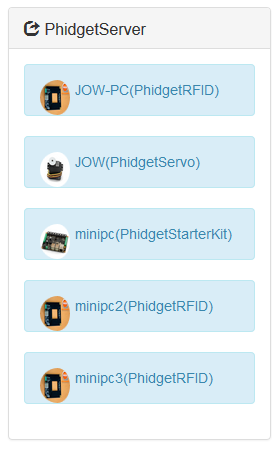
La communication en temps réel utilise la technologie WebSocket-SignalR.

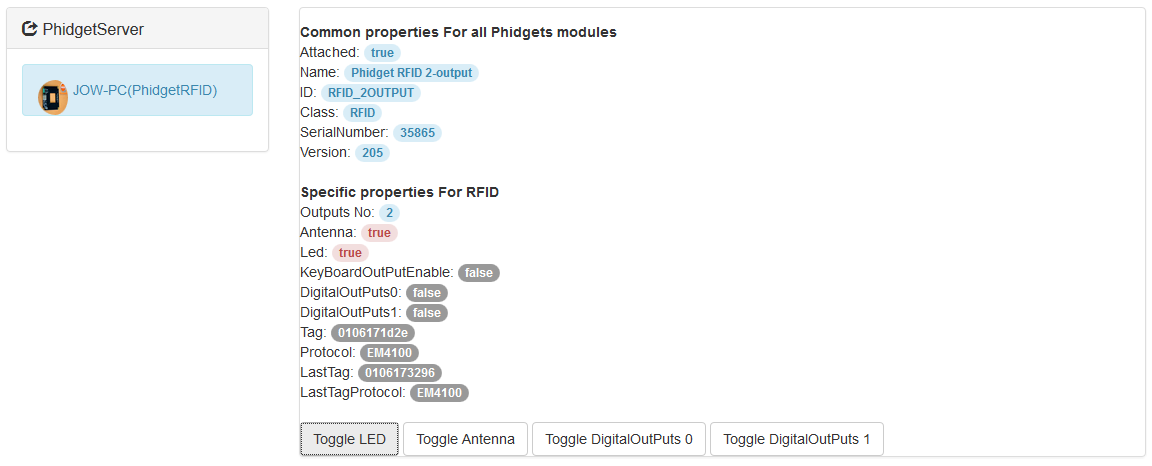


[PhidgetClient] et [PhidgetServer] font office de clients SignalR [SignalRClient]. Le serveur SignalR [PhidgetMiddlewareServerHubs] et le client [PhidgetClient] sont exécutés sur un serveur IIS 8 sous Windows Server 2012 R2. [PhidgetServer] est une application console s’exécutant sur un PC, un Raspberry Pi 2 B…

Le module Phidgets [PhidgetServer] attaché au port USB d’une machine peut être de même type ou de type différent sur les différentes machines accueillant le module Phidget.

L’utilisateur se voit proposer la liste des modules Phidget présents dans le réseau, identifiables par : NOM-MACHINE-ACCUEIL (TYPE-MODULE-PHIDGET).



En cliquant sur l’un des éléments de la liste, les données sont affichées en temps réel, et des contrôles (bouton…) permettent de piloter à distance actionneurs et sorties digitales. 

# Exploitation

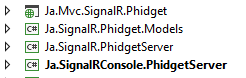


# Architecture



# Visual studio 2015 solution organization

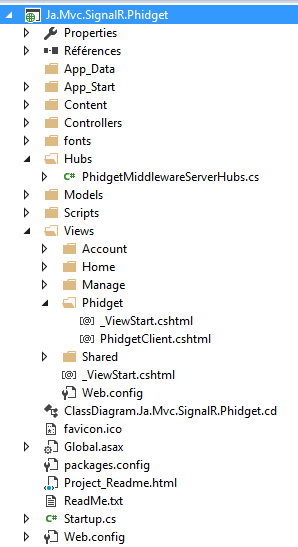
## Ja.Mvc.SignalR.Phidget solution projects.



## Ja.Mvc.SignalR.Phidget

ASP.NET MVC project hosting SignalR server Hub (Hubs/**PhidgetMiddlewareServer**.cs), and PhidgetClient (Views/Phidget/**PhidgetClient**.cshtml).

Depend on : Ja.SignalR.Phidget.Models

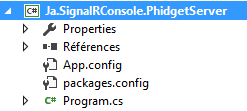


## Ja.SignalRConsole.PhidgetServer

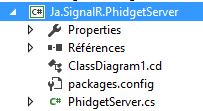
Windows console project hosting a PhidgetServer

Depend on : Ja.SignalR.Phidget.Models, Ja.SignalR.PhidgetServer

The goal is to launch a Task PhidgetServer included in Ja.SignalR.PhidgetServer

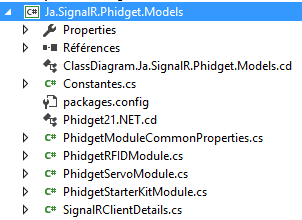


## Ja.SignalR.PhidgetServer



Class library project including the SignalR client code and the interaction with Phidget hardware

## Ja.SignalR.Phidget.Models



Class library project including the models for the PhidgetModules and the application DTO.

# Code

## Considération générales relatives au fonctionnement



7

6

5

4

3

2

1

L’image ci-dessus représente les actions déclenchées par les différents acteurs du processus.

1

Lorsqu’un client signalR [PhidgetClient] ou [PhidgetServer] démarre une connexion « **Start**() », la procédure évènementielle « **OnConnect**() » est appelée sur le server signalR [PhidgetMiddlewareServerHubs].

Aucune action particulière n’est exécutée côté serveur.

2

Après le démarrage de la connexion, le client doit systématiquement appeler « **AfterConnect**() »

En lui passant en paramètre un objet de type « **SignalRClientDetails** ».

L’action qui est exécutée dans « **AfterConect**() » dépend du type de client :

* Pour un [*PhidgetClient*] : le RPC « **show**() » de ce dernier est appelé, en lui passant la liste des [*PhidgetServer*] actuellement connectés afin que le client puisse alimenter sa liste des [*PhidgetServer*] actuellement connectés.
* Pour un [*PhidgetServer*] : le RPC « **OnNewPhidgetServerConnect**() » de tous les [*PhidgetClient*] est appelé en lui passant en paramètre l’objet de type « **SignalRClientDetails** » correspondant à ce [*PhidgetServer*].

Lorsqu’un utilisateur sélectionne sur un [PhidgetClient], un des [PhidgetServer] présent sur sa liste, la RPC « **OnPhidgetServerSelected**() » est appelée sur le server signalR [PhidgetMiddlewareServerHubs] en lui passant en paramètre l’objet correspondant au [PhidgetServer] sélectionné. Le serveur récupère les données par la RPC « **get**() » du client [PhidgetServer] en lui passant en paramètre l’ID du [PhidgetClient] ayant provoqué la demande. Dans cette RPC on appelle « **SendToPhidgetClient**() » du serveur en lui passant en paramètres : l’ID du [PhidgetClient] ayant provoqué la demande et les données extraites u Phidget. Le serveur n’a plus qu’à appeler la méthode « **show**() » du client concerné qui affichera les données correspondantes au [PhidgetServer] qui avait été sélectionné.

4

3

Lorsque l’utilisateur manipule un contrôle sur un [PhidgetClient] permettant de manipuler à distance un actionneur, la RPC « **SetByPhidgetClient**() » est appelée sur le server signalR [PhidgetMiddlewareServerHubs] en lui passant en paramètre l’objet correspondant au [PhidgetServer] manipulé, les données correspondantes au [PhidgetServer]. Le serveur appelle la RPC « **set**() » du client [PhidgetServer] en lui passant en paramètre les données. Dans cette RPC on appelle « **SendToPhidgetClients**() » du serveur, en lui passant en paramètre l’objet [PhidgetServer] ayant subi de la part d’un de ses [PhidgetClient] des modifications. La RPC appelée sur le serveur doit signaler les modifications apportées sur le [PhidgetServer] à tous les [PhidgetClient] ayant sélectionnés ce [PhidgetServer] en appelant leur RPC client « **show**() » avec les données modifiées.

5

Si l’utilisateur manipule directement les actionneurs, le client [PhidgetServer] sur lequel les manipulations sont effectuées doit pousser vers le serveur signalR [PhidgetMiddlewareServerHubs] les données en utilisant la RPC « **SendToPhidgetClients**() » en lui passant en paramètre l’objet [PhidgetServer] ayant subi des modifications. La RPC appelée sur le serveur doit signaler les modifications apportées sur le [PhidgetServer] à tous les [PhidgetClient] ayant sélectionnés ce [PhidgetServer] en appelant leur RPC client « **show**() » avec les données modifiées.

6

Lorsqu’un [PhidgetServer] se déconnecte suite à un arrêt de ce [PhidgetServer] ou suite à un « détachement » du module Phidget, la procédure évènementielle « **OnDisconnected**() » est automatiquement appelée côté serveur. Cette procédure doit signaler la fermeture de ce [PhidgetServer] à tous les [PhidgetClient] en appellant « **onPhidgetServerDisconnect**() » de tous les clients [PhidgetClient] afin que ce [PhidgetServer] soit supprimé de la liste des [PhidgetServer].

7

Lorsqu’un [PhidgetClient] se déconnecte suite à l’arrêt de ce [PhidgetClient], rien de particulier n’est à exécuter, la procédure évènementielle « **OnDisconnected**() » est automatiquement appelée côté serveur.

## Ja.SignalR.Phidget.Models

L’objet de transfert de données (Data Transfer Object ou DTO en anglais) est donné dans le diagramme de classe ci-dessous.





### DTO Phidget Module

Il est composé d’une classe abstraite contenant les propriétés communes à l’ensemble des modules Phidget et initialisés par le constructeur.

|  |
| --- |
| public PhidgetModuleProperties(dynamic phidgetModule)  {  this.Attached = phidgetModule.Attached;  this.Name = phidgetModule.Name;  this.ID = phidgetModule.ID.ToString();  this.Class = phidgetModule.Class.ToString();  this.SerialNumber = phidgetModule.SerialNumber;  this.Version = phidgetModule.Version;  } |

Une propriété « **PhidgetModuleType** » sera initialisée par « **PhidgetServer** ».

Enfin cette classe abstraite donne la signature de deux méthodes abstraites « **GetData** » et « **SetData** »,

|  |
| --- |
| public abstract PhidgetModuleCommonProperties GetData(PhidgetModuleCommonProperties phidgetData);  public abstract void SetData(dynamic phidgetData, dynamic phidgetModule); |

devant être implémentées par les classes concrètes.

On remarquera l’utilisation du type dynamic dont nous expliquerons la fonction un peu plus loin.

Le serveur SignalR « **PhidgetMiddlewareServerHubs** » maintien une liste des clients signalR connectés [PhidgetClient] et [PhidgetServer], les listes sont de type « **SignalRClientDetails** » cf. diagramme de classe précédent.

public abstract class PhidgetModuleProperties

{

public bool Attached { get; set; }

public string Name { get; set; }

public string ID { get; set; }

public string Class { get; set; }

public int SerialNumber { get; set; }

public int Version { get; set; }

public string PhidgetModuleType { get; set; }

public abstract PhidgetModuleProperties GetData(PhidgetModuleProperties phidgetData);

public abstract void SetData(dynamic phidgetData, dynamic phidgetModule);

public PhidgetModuleProperties(dynamic phidgetModule){}

}

public class PhidgetRFIDModule : PhidgetModuleProperties

{

public int Outputs { get; set; }

public bool LED { get; set; }

public bool Antenna { get; set; }

public bool KeyBoardOutPutEnable { get; set; }

public bool[] DigitalOutPuts { get; set; }

public string Tag { get; set; }

public bool TagPresent { get; set; }

public string LastTag { get; set; }

public string Protocol { get; set; }

public string LastTagProtocol { get; set; }

public PhidgetRFIDModule(RFID rfid) : base(rfid){}

public override PhidgetModuleProperties GetData(PhidgetModuleProperties phidgetData){}

public override void SetData(dynamic phidgetData, dynamic phidgetModule){}

}

public class PhidgetServoModule : PhidgetModuleProperties

{

public int ServoNumber { get; set; }

public int Count { get; set; }

public string[] Type { get; set; }

public double[] Position { get; set; }

//public double[] Degrees { get; set; }

//public double[] MinimumPulseWidth { get; set; }

//public double[] MaximumPulseWidth { get; set; }

public double[] PositionMin { get; set; }

public double[] PositionMax { get; set; }

public bool[] Engaged { get; set; }

public double[] PositionTrackBar { get; set; }

public PhidgetServoModule(Servo servo) : base(servo){}

public override PhidgetModuleProperties GetData(PhidgetModuleProperties phidgetData){}

public override void SetData(dynamic phidgetData, dynamic phidgetModule){}

}

public class PhidgetStarterKitModule : PhidgetModuleProperties

{

public int DigitalInputCount { get; set; }

public int DigitalOutputCount { get; set; }

public int AnalogInputCount { get; set; }

public bool Ratiometric { get; set; }

public bool[] DigitalInput { get; set; }

public bool[] DigitalOutput { get; set; }

public decimal[] AnalogInput { get; set; }

public int Deadband { get; set; }

public PhidgetStarterKitModule(InterfaceKit ifKit) : base(ifKit){}

public override PhidgetModuleProperties GetData(PhidgetModuleProperties phidgetData) {}

public override void SetData(dynamic phidgetData, dynamic phidgetModule) {}

}

### DTO SignalRClientDetails

public class SignalRClientDetails

{

public string ConnectionId { get; set; }

public string MachineName { get; set; }

public string IPAddress { get; set; }

public string ConnectionDateTime { get;}

public string PhidgetServerGroupAttachement { get; set; }

public string PhidgetModuleTypeAttached { get; set; }

public string SignalRClientTypeConnected { get; set; }

public SignalRClientDetails(){}

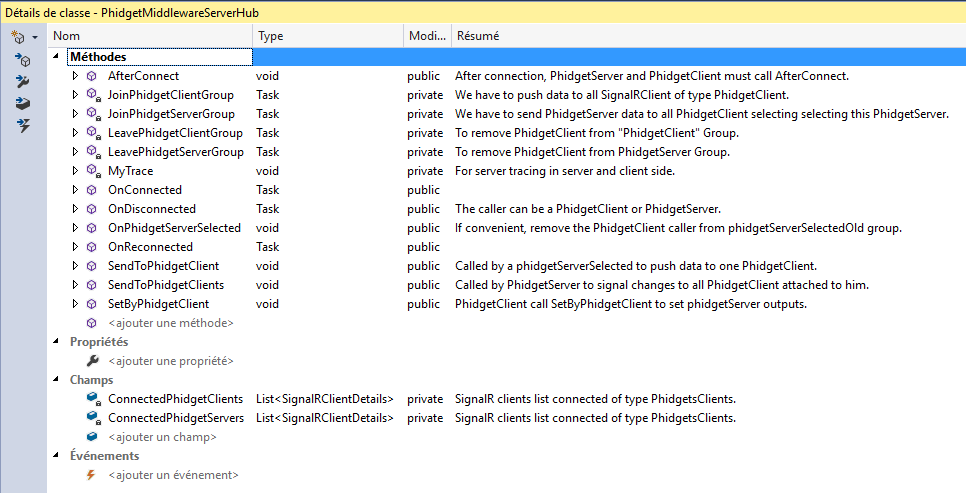
}

## Ja.Mvc.SignalR.Phidget



### SignalR server

Il est implémenté dans la classe « **PhidgetMiddlewareServerHubs** » cf. diagramme ci-dessus.



public class PhidgetMiddlewareServerHub : Hub

{

private static List<SignalRClientDetails> ConnectedPhidgetClients = new List<SignalRClientDetails>();

private static List<SignalRClientDetails> ConnectedPhidgetServers = new List<SignalRClientDetails>();

public override Task OnConnected()

public override Task OnDisconnected(bool stopCalled)

public void AfterConnect(SignalRClientDetails signalRClient)

public void OnPhidgetServerSelected(SignalRClientDetails phidgetServerSelected)

public void SendToPhidgetClient(string phidgetClientID, dynamic data)

public void SetByPhidgetClient(SignalRClientDetails phidgetServer, dynamic data)

public void SendToPhidgetClients(SignalRClientDetails phidgetServer, dynamic data)

private Task JoinPhidgetServerGroup(string phidgetServerGroupName)

private Task LeavePhidgetServerGroup(string phidgetServerGroupName)

private Task JoinPhidgetClientGroup()

private Task LeavePhidgetClientGroup()

private void MyTrace(string message)

}

### SignalR client

var phidgetClientHub;

var phidgetServerData;

var phidgetServerAttached;

var isButton = false;

$(function () {

});

function registerPhidgetClientMethods()

phidgetClientHub.client.onNewPhidgetServerConnect = function (phidgetServer)

phidgetClientHub.client.show = function (data) {

phidgetClientHub.client.onPhidgetServerDisconnect = function (phidgetServer)

function defineClass(value)

function testUndefined(value)

function loadservoknob()

function loadrotationknob()

function registerEvents()

$('#buttonToggleLED').click(function ()

$('#buttonToggleAntenna').click(function ()

$('#buttonToggleDigitalOutPuts0').click(function ()

$('#buttonToggleDigitalOutPuts1').click(function ()

$('#buttonServoPositionMin').click(function ()

$('#buttonServoLess').click(function ()

$('#buttonServoPlus').click(function ()

$('#buttonServoPositionMax').click(function ()

$('#buttonEngaged').click(function ()

$('#buttonRatiometric').click(function ()

$('#buttonDigitalOutPut1').click(function ()

$('#buttonDigitalOutPut2').click(function ()

$('#buttonDigitalOutPut3').click(function ()

$('#buttonDigitalOutPut4').click(function ()

$('#buttonDigitalOutPut5').click(function ()

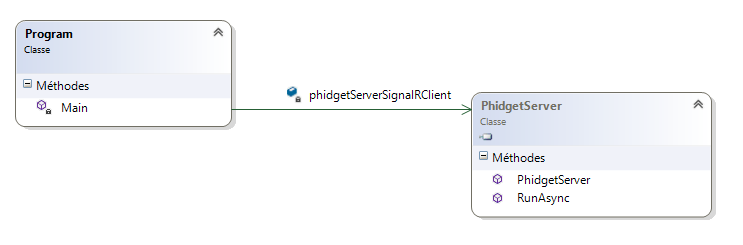
$('#buttonDigitalOutPut6').click(function ()

$('#buttonDigitalOutPut7').click(function ()

function AddPhidgetServer(phidgetServer)

function RemovePhidgetServer(phidgetServer)

## Ja.SignalRConsole.PhidgetServer



class Program

{

private static Ja.SignalR.PhidgetServer.PhidgetServer phidgetServerSignalRClient;

static void Main(string[] args)

{

string url = ConfigurationManager.AppSettings["url"];

//var writer = Console.Out;

var writer =

new StreamWriter(ConfigurationManager.AppSettings["PhidgetServerLog"]);

writer.AutoFlush = true;

phidgetServerSignalRClient =

new Ja.SignalR.PhidgetServer.PhidgetServer(

writer, ConfigurationManager.AppSettings["PhidgetModuleType"]);

phidgetServerSignalRClient.RunAsync(url).Wait();

Console.ReadKey();

}

}

## Ja.SignalR.PhidgetServer



public class PhidgetServer

{

private TextWriter \_traceWriter;

private HubConnection \_hubConnection;

private IHubProxy \_phidgetServerHubProxy;

private static PhidgetModuleProperties \_phidgetData;

private static SignalRClientDetails \_phidgetServer;

private static string \_phidgetModuleType;

private static RFID rfid; //Declare an RFID object

private static Servo servo; //Declare an Servo object

private static InterfaceKit ifKit; //Declare an InterfaceKit object

public PhidgetServer(TextWriter traceWriter, string phidgetModuleType)

public async Task RunAsync(string url)

private async Task RunPhidgetServerAsync(string url)

private void Init()

private void Terminate()

private void RegisterPhidgetServerMethods(){

\_phidgetServerHubProxy.On<string>("get", async (string x) =>{});

\_phidgetServerHubProxy.On<dynamic>("set", async (dynamic x) =>{});

}

private void InitPhidgetHardware()

private void ClosePhidgetHardware()

private void GetPhidgetServerData()

private void SetPhidgetServerData(dynamic data)

private void InitPhidgetServerInformations()

private string GetMyIpAddress()

private void rfidInitPhidgetHardware()

private void rfid\_Attach(object sender, AttachEventArgs e)

private void rfid\_Detach(object sender, DetachEventArgs e)

private void rfid\_Tag(object sender, TagEventArgs e)

private void rfid\_TagLost(object sender, TagEventArgs e)

private void rfidClosePhidgetHardware()

private void servoInitPhidgetHardware()

private void servoClosePhidgetHardware()

private void servo\_Attach(object sender, AttachEventArgs e)

private void servo\_Error(object sender, Phidgets.Events.ErrorEventArgs e)

private void ifKitInitPhidgetHardware()

private void ifKit\_Attach(object sender, AttachEventArgs e)

private void ifKit\_Detach(object sender, DetachEventArgs e)

private void ifKit\_Error(object sender, Phidgets.Events.ErrorEventArgs e)

private void ifKit\_InputChange(object sender, InputChangeEventArgs e)

private void ifKit\_OutputChange(object sender, OutputChangeEventArgs e)

private void ifKit\_SensorChange(object sender, SensorChangeEventArgs e)

private void ifKitClosePhidgetHardware()

}

# Glossaire:

---------

- **PhidgetClient** = (Desktop, Smartphone, Tablet)

ASP.NET MVC page from ASP.NET MVC application PhidgetMiddlewareServer + SignalR client with proxy hub + Javascript

- **PhidgetServer** = (Windows 8, Windows 8.1, Windows 10)

Windows .NET application with PhidgetModule attached + SignalR client + C#

- **PhidgetMiddlewareServer** = (Windows server 2012 R2, Windows 8, Windows 8.1, Windows 10)

ASP.NET MVC IIS 8 application + SignalR hub server + C#

- **PhidgetModule** = Phidget product attached to a USB port PhidgetServer

ex: 2003\_5 - Phidget Starter Kit #1, 1024\_0 - PhidgetRFID Read-Write , 1066\_1 - PhidgetAdvancedServo 1-Motor

- PhidgetSensor = Phidget product with digital/analog input/output

ex: 1106 - Force Sensor, 1109 - Rotation Sensor, 1129 - Touch Sensor, 1112 - Slider 60 Sensor, 1127 – Precision Light Sensor

- **SignalRClient** = PhidgetClient or PhidgetServer

- **PhidgetClienUser** = User who manipulate PhidgetClient UI.