UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Projecte de desenvolupament d'un sistema de posicionament automàtic per a la caracterització d'antenes

Document:
Annexos

Autor:
Pol Monreal i Mira

Director:
Raúl Fernández Garcia

Titulació:
Màster Universitari en Enginyeria Industrial

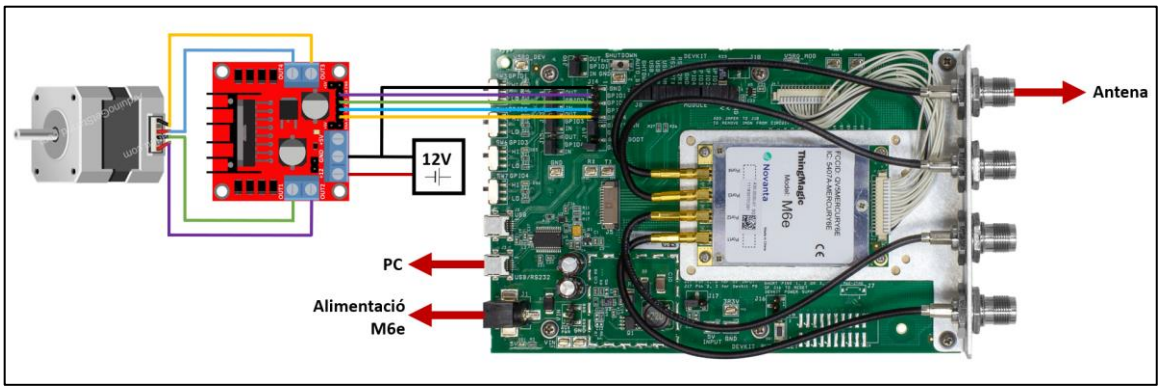
Convocatòria:
Primavera 2022

TREBALL DE FI D'ESTUDIS

Índex

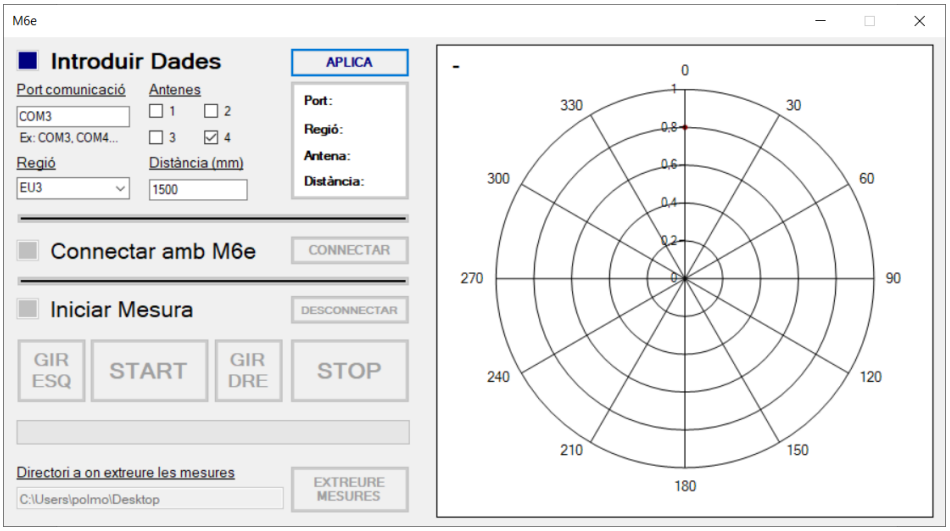
1 Annex A: Manual instruccions sistema desenvolupat

1) Connexions M6e



- Alimentar M6e (Adaptador)
- Connectar M6e a PC (micro USB)
- Alimentar Driver (Font DC 12V i Intensitat Max)

2) Obrir Aplicació (M6e_APP.exe)



3) Introduir dades

- Port Comunicació: Obrir administrador dispositius i comprovar port COM utilitzat (cal haver connectat el M6e al PC)
- Regió: Seleccionar una de les opcions del desplegable
- Antenes: Seleccionar com a mínim una (al laboratori l'antena utilitzada es la 4)
- Distancia: Ha de ser superior a 0
- Clicar **APLICA** (si hi ha alguna dada incorrecte saltarà un missatge)

4) Connectar amb M6e

- Clicar **CONNECTAR** (si hi ha algun error saltarà un missatge)

5) Ajustar posició motor (opcional)

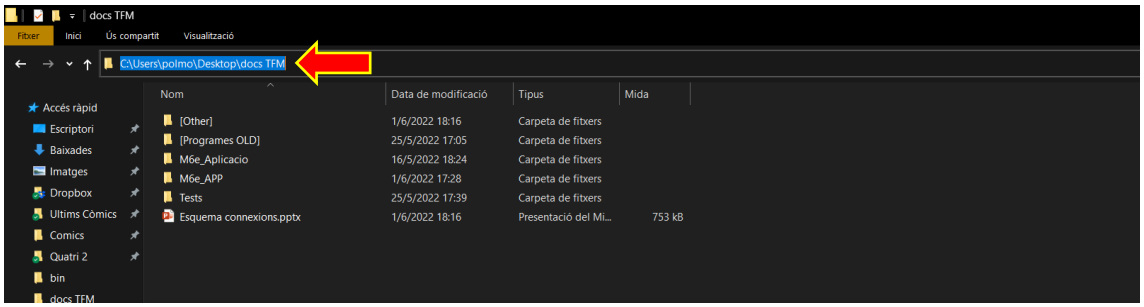
- Es pot ajustar la posició del motor amb els botons **GIR ESQ** i **GIR DRE**
- Un cop ajustada la posició desitjada podem iniciar mesura

6) Iniciar Mesura

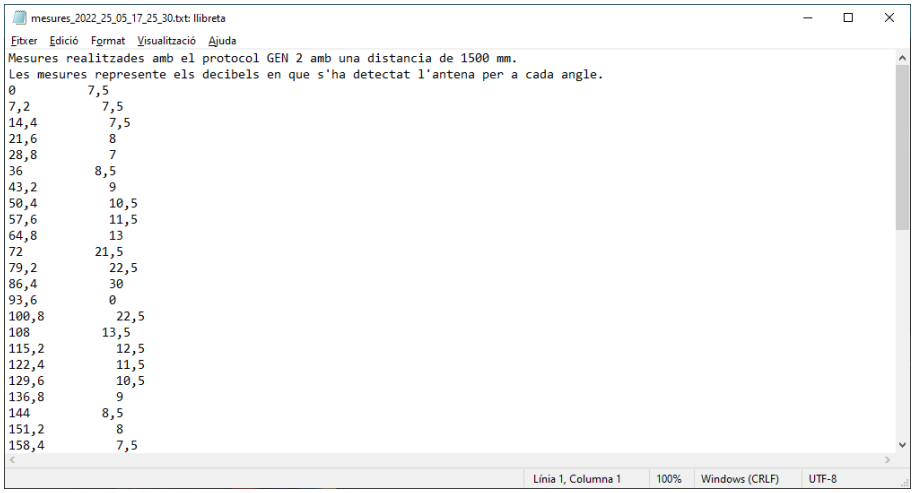
- Clicar **START**
- Podem veure com es genera un gràfic en temps real
- Esperem a que finalitzin el procés de mesura (a la part inferior veiem una barra de progrés que indica les mesures realitzades i les restants)
- Si volem interrompre la mesura en qualsevol moment, clicar **STOP**

7) Extreure mesures (opcional)

- Un cop finalitzat el procés de mesura (o s'ha clicat STOP) podem extreure les dades de l'assaig
- Primer cal introduir el directori d'on volem extreure les dades (obrir l'explorador de caretes i copiar-lo directament)



- Clicar **EXTREURE DADES**
- Es generarà un fitxer txt amb l'angle de cada una de les posicions mesurades juntament amb la potencia de detecció en decibels.



8) Desconnectar

- Un cop finalitzat l'assaig cliquem **DESCONECTAR** per a tornar a l'estat inicial, des d'on podem tornar a començar amb el procés

2 Annex B: Codi software de control

```
// Declaració del les referencies a utilitzar

using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using System.Drawing;
using System.Windows.Forms;
using ThingMagic;
using System.IO;

namespace ProbaAplica1
{
public partial class M6E : Form
{

public M6E()
{
InitializeComponent();
chart.Series["F1"].Points.AddXY(0, 0.8); //Iniciem el programa di-
buixant el gràfic "net"
}


//Declarem variables globals
int ACCIO;
int ESTAT;
int POS;
int[] antena;
int dades;
string port;
string llista;
bool conectat;
bool detectat;
bool STOP;
bool bucle;
int potencia;
int distancia;
double calcul_formula;
List<double> llista_mesures = new List<double>();
TagReadData[] tagReads;

// Subprograma ACCIO_APP: Executa les diferents ordres en funció
de la variable "ACCIO"
private async void ACCIO_APP()
{
switch (ACCIO)
{
case 0: //S'ha clicat aplica dades
{
//"Netegem" variables
dades = 0;
distancia = 0;
antena = null;
```

```
port = null;
llista = null;
lbl_port.Text = null;
lbl_Regio.Text = null;
lbl_antena.Text = null;
lbl_dist.Text = null;

//Comprovem que s'hagi introduït un port
if (txt_port_com.Text != "")
{
port = "tmr:///" + txt_port_com.Text;
lbl_port.Text = txt_port_com.Text;
dades++;
}
else MessageBox.Show("Cal introduir el port de comunicacions");

//Comprovem que s'hagi introduït una regió
if (select_regio.Text != "")
{
lbl_Regio.Text = select_regio.Text;
dades++;
}
else MessageBox.Show("Cal introduir una regió");

//Comprovem que s'hagi introduït com a mínim 1 antena
if (cbx_ant1.Checked)
{
if ((cbx_ant2.Checked) || (cbx_ant3.Checked) || (cbx_ant4.Chec-
ked)) llista = llista + "1,";
else llista = llista + "1";
}
if (cbx_ant2.Checked)
{
if ((cbx_ant3.Checked) || (cbx_ant4.Checked)) llista = llista +
"2,";
else llista = llista + "2";
}
if (cbx_ant3.Checked)
{
if ((cbx_ant4.Checked)) llista = llista + "3,";
else llista = llista + "3";
}
if (cbx_ant4.Checked)
{
llista = llista + "4";
}
if (llista != null)
{
antena = Array.ConvertAll<string, int>(llista.Split(','),
int.Parse); //Creem el vector que necessita el M6e amb les antenes
seleccionades
lbl_antena.Text = llista;
dades++;
}
else MessageBox.Show("Cal marcar com a mínim 1 antena");
```

```
//Comprovem que la distancia introduïda sigui adient
try
{
distancia = Int32.Parse(txt_dist.Text);
if (distancia > 0)
{
lbl_dist.Text = txt_dist.Text;
dades++;
}
else MessageBox.Show("La distancia ha de ser superior a 0 mm");
}
catch (Exception ex1)
{
if (txt_dist.Text == "") MessageBox.Show("Cal introduir una dis-
tancia");
else MessageBox.Show("Distancia introduïda en format incorrecte,
enter superior a 0." + ex1.Message);
}

//Comprovem si s'han introduït totes les dades necessàries
if (dades == 4)
{
ESTAT = 1;
ESTAT_APP();
}
}
break;

case 1: //S'ha clicat connectar amb m6e
{
ESTAT = 5;
ESTAT_APP();

STOP = false;
bucle = false;
conectat = false;
try
{
using (Reader r = Reader.Create(port))//Creem objecte reader
{
r.Connect();//Connectem amb el M6e

//Ajustem ports GPIO a la posició inicial
int[] output = new int[] { 1, 2, 3, 4 };
r.ParamSet("/reader/gpio/outputList", output);
POS = 1;
r.GpoSet(CODIFICA(POS));
await Task.Run(() => System.Threading.Thread.Sleep(250));
r.GpoSet(CODIFICA(0));

//Comprovem si detecta antena
if (r.isAntDetectEnabled(antena))
{
MessageBox.Show("Antena no detectada");
```

```
}
else conectat = true;

//Establim regió
if (select_regio.Text == "NA") r.ParamSet("/reader/region/id", Re-
ader.Region.NA);
else if (select_regio.Text == "EU3") r.ParamSet("/reader/re-
gion/id", Reader.Region.EU3);
else if (select_regio.Text == "KR2") r.ParamSet("/reader/re-
gion/id", Reader.Region.KR2);
else if (select_regio.Text == "PRC") r.ParamSet("/reader/re-
gion/id", Reader.Region.PRC);
else if (select_regio.Text == "AU") r.ParamSet("/reader/re-
gion/id", Reader.Region.AU);
else if (select_regio.Text == "NZ") r.ParamSet("/reader/re-
gion/id", Reader.Region.NZ);
else if (select_regio.Text == "OPEN") r.ParamSet("/reader/re-
gion/id", Reader.Region.OPEN);
else
{
MessageBox.Show("Regio no compatible");
conectat = false;
}
}
}
catch (Exception error)
{
MessageBox.Show(error.Message);
}

//Comprovem si la connexió ha estat efectiva
if (conectat)
{
ESTAT = 2;
ESTAT_APP();
}
else
{
ESTAT = 0;
ESTAT_APP();
}
}
break;

case 2: //Gir motor Esquerra
{
ESTAT = 5;
ESTAT_APP();

using (Reader r = Reader.Create(port))//Creem objecte reader
{
r.Connect();//Connectem amb m6e

//Disminuïm la posició en 4 unitats
for (int i = 0; i < 4; i++)
```

```
{
if (POS > 1) POS--;
else POS = 4;
r.GpoSet(CODIFICA(POS));
await Task.Run(() => System.Threading.Thread.Sleep(250));
}
r.GpoSet(CODIFICA(0));
}

ESTAT = 2;
ESTAT_APP();
}
break;

case 3: //Gir motor Dreta
{
ESTAT = 5;
ESTAT_APP();

using (Reader r = Reader.Create(port))//Creem objecte reader
{
r.Connect();//Connectem amb m6e

//Augmentem la posició en 4 unitats
for (int i = 0; i < 4; i++)
{
if (POS < 4) POS++;
else POS = 1;
r.GpoSet(CODIFICA(POS));
await Task.Run(() => System.Threading.Thread.Sleep(250));
}
r.GpoSet(CODIFICA(0));
}

ESTAT = 2;
ESTAT_APP();
}
break;

case 4: //S'ha clicat inicia mesura
{
ESTAT = 3;
ESTAT_APP();

chart.Series["F1"].Points.Clear();//Netegem grafic abans de dibui-
xar
BarraP.Value = 0;//Netegem barra progres
STOP = false; //Desactivem bool stop
bucle = true; //Activem bool bucle
llista_mesures.Clear(); //Netegem llista mesures

//////Comencem assaig
using (Reader r = Reader.Create(port))//Creem objecte reader
{
```

```
r.Connect();//Connectem amb m6e
r.ParamSet("/reader/read/plan", new SimpleReadPlan(antena, TagPro-
tocol.GEN2, null, null, 1000)); //Ajustem el protocol de mesura

//Realitzem la mesura 51 vegades (1 volta + posició inicial)
for (int i = 0; i < 51 & !STOP; i++)
{
detectat = false;
potencia = 500; //Ajustem potencia inicial a 5 dB, la mínima que
permet el M6e

//Mentre no es detecti el TAG, augmentem la potencia 0,5 db cada
cop fins als 31 dB màxims que permet el M6e
while (!detectat & potencia < 3100)
{
r.ParamSet("/reader/radio/readPower", potencia);
await Task.Run(() => tagReads = r.Read(20)); // Realitzem la me-
sura durant 20 ms
if (tagReads.Length > 0) detectat = true;
else potencia = potencia + 50;
}

//Si després de realitzar la mesura s'ha detectat el TAG, es gra-
fica el punt i s'afegeix la potencia de detecció a la llista de
mesures
if (detectat)
{
calcul_formula = potencia;
calcul_formula = calcul_formula / 100;
lbl_MESURA.Text = calcul_formula.ToString() + " dBm";
chart.Series["F1"].Points.AddXY(i * 360 / 50, 5 / calcul_formula);
//Grafiquem la divisió de 5 dB per la potencia de detecció
}
//Si no ha detectat el TAG, grafiquem un 0 i l'afegim a la llista
else
{
lbl_MESURA.Text = "fora de rang";
calcul_formula = 0;
chart.Series["F1"].Points.AddXY(i * 360 / 50, 0);
}

//Després de cada mesura augmentem la posició del motor en 4 uni-
tats (200 posicions del motor entre 4 equival a realitzar 50 mesu-
res)
for (int a = 0; a < 4;)
{
if (POS < 4) POS++;
else POS = 1;
r.GpoSet(CODIFICA(POS));
await Task.Run(() => System.Threading.Thread.Sleep(250));
a++;
}
r.GpoSet(CODIFICA(0));
await Task.Run(() => System.Threading.Thread.Sleep(250));
```

```
llista_mesures.Add(calcul_formula);
BarraP.Value = 4 * i;
}
POS = 0;
r.GpoSet(CODIFICA(POS)); //Un cop finalitzades les mesures desac-
tivem els ports GPIO
lbl_MESURA.Text = "-";
}
bucle = false;
ESTAT = 4;
ESTAT_APP();
}
break;

case 5: // S'ha clicat extreure dades
{
//Primer comprovem que el directori on extraurem les dades exis-
teix
if (Directory.Exists(txt_direccio_exp.Text))
{
string directori_nom_arxiu = txt_direccio_exp.Text + "//mesures_"
+ DateTime.Now.ToString("yyyy_dd_MM_HH_mm_ss") + ".txt"; //Afegim
la data i hora en que s'ha realitzat l'assaig al arxiu amb les da-
des
TextWriter exportar = new StreamWriter(directori_nom_arxiu);
//En l'arxiu a exportar primerament afegim un comentari explicant
les condicions de l'assaig i el format del resultat
exportar.WriteLine("Mesures realitzades amb el protocol GEN 2 amb
una distancia de " + distancia + " mm.");
exportar.WriteLine("Les mesures representen els decibels en que
s'ha detectat l'antena per a cada angle.");
//Afegim a l'arxiu cada una de les mesures realitzades juntament
amb l'angle corresponent
for (int i = 0; i < llista_mesures.Count; i++)
{
double angle = i * 7.2;
exportar.WriteLine(angle + "          " + llista_mesures[i]);
}
exportar.Close();
}
else MessageBox.Show("El directori no existeix");
}
break;

case 6: // S'ha clicat desconnectar de m6e
{

ESTAT = 5;
ESTAT_APP();

chart.Series["F1"].Points.Clear(); //Netegem gràfic
chart.Series["F1"].Points.AddXY(0, 0.8);//Dibuixem gràfic "net"
BarraP.Value = 0;//Netegem barra progres

//Ens assegurem de que els ports GPIO estiguin desactivats
```

```
POS = 0;
using (Reader r = Reader.Create(port))//Creem objecte reader
{
r.Connect();// connectem amb m6e
r.GpoSet(CODIFICA(POS));
}

ESTAT = 0;
ESTAT_APP();
}
break;

default:
break;
}
}

// Subprograma ESTAT_APP: Habilita o deshabilita els elements de
la aplicació (botons clicables, textos modificables...) en funció
de la variable "ESTAT"
private void ESTAT_APP()
{
switch (ESTAT)
{
case 0: // S'han d'introduir dades i clicar aplica dades
{
Ind_dades.BackColor = Color.Navy;
Ind_conect.BackColor = Color.Silver;
Ind_mesura.BackColor = Color.Silver;

lbl_port.Text = null;
lbl_Regio.Text = null;
lbl_antena.Text = null;
lbl_dist.Text = null;

txt_port_com.Enabled = true;
txt_dist.Enabled = true;
txt_direccio_exp.Enabled = false;

select_regio.Enabled = true;

cbx_ant1.Enabled = true;
cbx_ant2.Enabled = true;
cbx_ant3.Enabled = true;
cbx_ant4.Enabled = true;

btn_aplica.Enabled = true;
btn_conect.Enabled = false;
btn_desconect.Enabled = false;
btn_start.Enabled = false;
btn_gir_E.Enabled = false;
btn_gir_D.Enabled = false;
btn_stop.Enabled = false;
btn_extract.Enabled = false;
}
```

```
break;

case 1: // S'han verificat les dades i cal clicar connectar
{
Ind_dades.BackColor = Color.Silver;
Ind_conect.BackColor = Color.Navy;
Ind_mesura.BackColor = Color.Silver;

txt_port_com.Enabled = false;
txt_dist.Enabled = false;
txt_direccio_exp.Enabled = false;

select_regio.Enabled = false;

cbx_ant1.Enabled = false;
cbx_ant2.Enabled = false;
cbx_ant3.Enabled = false;
cbx_ant4.Enabled = false;

btn_aplica.Enabled = false;
btn_conect.Enabled = true;
btn_desconect.Enabled = false;
btn_start.Enabled = false;
btn_gir_E.Enabled = false;
btn_gir_D.Enabled = false;
btn_stop.Enabled = false;
btn_extract.Enabled = false;
}
break;

case 2: // S'ha conectat amb el M6e, cal iniciar mesura, ajustar
angle motor o desconnectar
{
Ind_dades.BackColor = Color.Silver;
Ind_conect.BackColor = Color.Silver;
Ind_mesura.BackColor = Color.Navy;


txt_port_com.Enabled = false;
txt_dist.Enabled = false;
txt_direccio_exp.Enabled = false;


select_regio.Enabled = false;


cbx_ant1.Enabled = false;
cbx_ant2.Enabled = false;
cbx_ant3.Enabled = false;
cbx_ant4.Enabled = false;
```

```
btn_aplica.Enabled = false;
btn_conect.Enabled = false;
btn_desconect.Enabled = true;
btn_start.Enabled = true;
btn_gir_E.Enabled = true;
btn_gir_D.Enabled = true;
btn_stop.Enabled = false;
btn_extract.Enabled = false;
}
break;

case 3: // S'està realitzant mesura
{
Ind_dades.BackColor = Color.Silver;
Ind_conect.BackColor = Color.Silver;
Ind_mesura.BackColor = Color.Navy;

txt_port_com.Enabled = false;
txt_dist.Enabled = false;
txt_direccio_exp.Enabled = false;

select_regio.Enabled = false;

cbx_ant1.Enabled = false;
cbx_ant2.Enabled = false;
cbx_ant3.Enabled = false;
cbx_ant4.Enabled = false;

btn_aplica.Enabled = false;
btn_conect.Enabled = false;
btn_desconect.Enabled = false;
btn_start.Enabled = false;
btn_gir_E.Enabled = false;
btn_gir_D.Enabled = false;
btn_stop.Enabled = true;
btn_extract.Enabled = false;
}
```

```
break;

case 4: // Ha finalitzat la mesura o s'ha clicat boto STOP, cal
extreure dades o desconnectar
{
Ind_dades.BackColor = Color.Silver;
Ind_conect.BackColor = Color.Silver;
Ind_mesura.BackColor = Color.Navy;

txt_port_com.Enabled = false;
txt_dist.Enabled = false;
txt_direccio_exp.Enabled = true;

select_regio.Enabled = false;

cbx_ant1.Enabled = false;
cbx_ant2.Enabled = false;
cbx_ant3.Enabled = false;
cbx_ant4.Enabled = false;

btn_aplica.Enabled = false;
btn_conect.Enabled = false;
btn_desconect.Enabled = true;
btn_start.Enabled = false;
btn_gir_E.Enabled = false;
btn_gir_D.Enabled = false;
btn_stop.Enabled = false;
btn_extract.Enabled = true;
}
break;

case 5: // No es pot clicar res, el programa esta treballant
{
Ind_dades.BackColor = Color.Silver;
Ind_conect.BackColor = Color.Silver;
Ind_mesura.BackColor = Color.Navy;
```

```
txt_port_com.Enabled = false;
txt_dist.Enabled = false;
txt_direccio_exp.Enabled = false;

select_regio.Enabled = false;

cbx_ant1.Enabled = false;
cbx_ant2.Enabled = false;
cbx_ant3.Enabled = false;
cbx_ant4.Enabled = false;

btn_aplica.Enabled = false;
btn_conect.Enabled = false;
btn_desconect.Enabled = false;
btn_start.Enabled = false;
btn_gir_E.Enabled = false;
btn_gir_D.Enabled = false;
btn_stop.Enabled = false;
btn_extract.Enabled = false;
}
break;

default:
break;
}
}


// Accions a executar quan es clica un boto

public void btn_aplica_Click(object sender, EventArgs e)
{
ACCIO = 0;
ACCIO_APP();
}// boto aplicar dades

private void btn_conect_Click(object sender, EventArgs e)
```

```
{
ACCIO = 1;
ACCIO_APP();
}// boto connectar amb m6e
private void btn_gir_E_Click(object sender, EventArgs e)
{
ACCIO = 2;
ACCIO_APP();
}// boto gir ESQ

private void btn_gir_D_Click(object sender, EventArgs e)// boto gir
DRET
{
ACCIO = 3;
ACCIO_APP();
}// boto gir DRE

private void btn_start_Click(object sender, EventArgs e)
{
ACCIO = 4;
ACCIO_APP();
}// boto start

private void btn_extract_Click(object sender, EventArgs e)
{
ACCIO = 5;
ACCIO_APP();
}// boto extreure mesures

private void btn_desconect_Click(object sender, EventArgs e)
{
ACCIO = 6;
ACCIO_APP();
}// boto desconnecta


//El boto STOP es especial, pot interrompre el bucle de mesures
```

```
private async void btn_STOP_Click(object sender, EventArgs e)
{
ESTAT = 5;
ESTAT_APP();

STOP = true;
while (bucle) await Task.Run(() => System.Threading.Thread.Sleep(1000));

lbl_MESURA.Text = "-";
ESTAT = 4;
ESTAT_APP();
}


//Subprograma CODIFICA: entrem amb posició (1,2,3,4 o 0) i retorna variable "gps" necessària per a executar set GPIO
private static GpioPin[] CODIFICA(int POS)
{
List<GpioPin> gps = new List<GpioPin>();

if (POS == 0)
{
GpioPin P1 = new GpioPin(1, false);
GpioPin P2 = new GpioPin(2, false);
GpioPin P3 = new GpioPin(3, false);
GpioPin P4 = new GpioPin(4, false);
gps.Add(P1);
gps.Add(P2);
gps.Add(P3);
gps.Add(P4);
}

if (POS == 1)
{
GpioPin P1 = new GpioPin(1, true);
GpioPin P2 = new GpioPin(2, false);
GpioPin P3 = new GpioPin(3, true);
```

```
GpioPin P4 = new GpioPin(4, false);
gps.Add(P1);
gps.Add(P2);
gps.Add(P3);
gps.Add(P4);
}

if (POS == 2)
{
GpioPin P1 = new GpioPin(1, false);
GpioPin P2 = new GpioPin(2, true);
GpioPin P3 = new GpioPin(3, true);
GpioPin P4 = new GpioPin(4, false);
gps.Add(P1);
gps.Add(P2);
gps.Add(P3);
gps.Add(P4);
}

if (POS == 3)
{
GpioPin P1 = new GpioPin(1, false);
GpioPin P2 = new GpioPin(2, true);
GpioPin P3 = new GpioPin(3, false);
GpioPin P4 = new GpioPin(4, true);
gps.Add(P1);
gps.Add(P2);
gps.Add(P3);
gps.Add(P4);
}

if (POS == 4)
{
GpioPin P1 = new GpioPin(1, true);
GpioPin P2 = new GpioPin(2, false);
GpioPin P3 = new GpioPin(3, false);
GpioPin P4 = new GpioPin(4, true);
```

```
gps.Add(P1);
gps.Add(P2);
gps.Add(P3);
gps.Add(P4);
}

return gps.ToArray();
}
}
}
```