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:
Memòria

Autor:
Pol Monreal i Mira

Director:
Raúl Fernández Garcia

Titulació:
Màster Universitari en Enginyeria Industrial

Convocatòria:
Primavera 2022

TREBALL DE FI D'ESTUDIS

Resum

L'objectiu d'aquest projecte és el desenvolupament d'un sistema de posicionament automàtic per a realitzar assajos de caracterització d'antenes RFID, per al qual s'ha utilitzat el kit de desenvolupament M6e de ThingMagic.

El kit M6e és un mòdul integrat que, entre altres, permet mesurar la potencia de detecció d'antenes RFID i que compta amb 4 ports GPIO que permeten, mitjançant senyals digitals que processa el driver L298N Motor Driver Module, ajustar la posició del motor pas a pas Longruner Nema 17. Amb aquests elements el sistema permet realitzar la mesura de la potencia de detecció de les antenes en 50 posicions separades per 7,2º (7,2º * 50 = 360º), per a obtenir el seu diagrama de radiació.

El projecte inclou el disseny i el muntatge del sistema de posicionament, tant el la part elèctrica com el la de la seva estructura mecànica, així com el disseny i programació del seu software de control i el d'una aplicació interfície.

Aquesta aplicació interfície permet als usuraris introduir les dades de l'assaig (regió, distancia entre el sistema de mesura i l'antena a mesurar, port de comunicació...), ajustar manualment la posició inicial del motor, iniciar i aturar els assajos de mesura, obtenir el diagrama de radiació en temps real i extreure les dades del assaig en un document txt.

Per al desenvolupament d'aquest software de control i de la aplicació interfície, s'ha utilitzat el programa Visual Studio amb el llenguatge de programació C#.

Finalment el projecte també compta amb una fase d'assajos reals amb diferents antenes RFID per a verificar el correcte funcionament del sistema desenvolupat.

Abstract

The goal of this project is to develop an automatic positioning system to conduct RFID antenna characterization tests, for which has been used the ThingMagic M6e development kit.

The M6e kit is an integrated module that, among other things, allows to measure the detection power of RFID antennas and has 4 GPIO ports that allow, using digital signals processed by the L298N Motor Driver Module, adjust the position of the stepper motor Longruner Nema 17. With these elements the system allows the measurement of the detection power of the antennas in 50 positions separated by 7.2º (7.2º * 50 = 360º), to obtain its radiation pattern.

The project includes the design and assembly of the positioning system, both the electrics and the mechanical structure, as well as the design and programming of its control software and an interface application.

This interface application allows users to enter the test data (region, distance between the measuring system and the antenna to be measured, communication port...), manually adjust the initial position of the motor, start and stop the tests measurement, obtain the real-time radiation pattern and extract the test data in a txt document.

For the development of this control software and the interface application, the Visual Studio program with the C # programming language has been used.

Finally, the project also includes a phase of tests with different RFID antennas to verify the correct operation of the developed system.

Índex

Índex de figures

1. Introducció

1.1 Objecte

L'objecte d'aquest projecte és el desenvolupament d'un sistema de posicionament automàtic per a realitzar assajos de caracterització d'antenes RFID.

1.2 Abast

Activitats a realitzar:

Dissenyar i construir un sistema de posicionament d'antenes RFID que permeti ajustar-ne la posició i obtenir la potencia de detecció en cada una utilitzant el kit de desenvolupament M6e. Aquesta activitat inclourà tant el disseny i ensamblatge de la part elèctrica, dimensionant i seleccionant els components, com el de la seva estructura mecànica.

Dissenyar i programar el software per al control del sistema de posicionament descrit. Per a aquesta activitat serà necessària una recerca d'informació sobre el kit de desenvolupament M6e per a entendre el funcionament de la seva programació.

Dissenyar i programar una aplicació que permeti als usuaris interaccionar i controlar el sistema descrit de tal forma que es pugui obtenir com a resultat final tant el diagrama de radiació com les dades resultants dels assajos.

Per a aquestes darrers dues tasques, s'utilitzarà el programa Visual Studio, pel que serà necessària tant una recerca d'informació com una formació practica, per a aprendre a utilitzar-lo.

Finalment, també s'inclou a l'abast la realització d'assajos amb diferents antenes RFID per a verificar el correcte funcionament del sistema un cop finalitzada la seva construcció.

1.3 Requeriments

El sistema de mesura sobre el que gira aquest TFM haurà de complir amb les següents especificacions:

- Utilitzant el kit de desenvolupament M6e, s'ha de poder obtenir la potencia de detecció d'antenes RFID.

- Utilitzant els ports GPIO del kit de desenvolupament M6e, s'ha de poder controlar un motor pas a pas per a ajustar la posició d'antenes RFID.

- El sistema ha de realitzar 50 mesures separades per 7,2º (7,2º * 50 = 360)per a obtenir el diagrama de radiació complert

- L'usuari ha de poder ajustar, a traves de la aplicació interfície, les dades de l'assaig (regió, port de comunicacions, antena utilitzada i distancia entre el sistema de mesura i l'antena a mesurar).

- L'usuari ha de poder ajustar, a traves de la aplicació interfície, la posició inicial del motor controlat.

- L'usuari ha de poder iniciar els assajos i comptar amb un boto de parada (si sorgís algun problema s'ha de poder aturar l'assaig en qualsevol moment).

- L'aplicació interfície ha de generar el diagrama de radiació de l'antena mesurada.

- L'aplicació interfície ha de poder generar un fitxer d'on extreure les dades dels assajos.

- L'aplicació ha de permetre realitzar múltiples assajos consecutius (al finalitzar un assaig s'ha de poder iniciar un de nou sense haver de reiniciar el sistema).

- El sistema de mesura ha de funcionar correctament davant de diferents antenes.

1.4 Justificació

La necessitat d'aquest projecte, va sorgir principalment per la feina que es realitza al departament de d'enginyeria electrònica de l'Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa de la UPC, on estan dissenyant antenes RFID per a teixits on, a diferencia de les tecnologies actuals on simplement s'integren aquestes antenes a la roba, les pròpies antenes estan fetes de teixit.

En tant que un dels principals processos per a verificar el correcte disseny i funcionament d'aquestes antes és el d'obtenir el seu diagrama de radiació, procés per al qual es necessita determinar la potencia de detecció en diferents posicions, el departament va adquirir el kit de desenvolupament M6e, el qual esta format principalment d'un mòdul integrat que controla una antena detectora. Aquest kit, també inclou un programa basic per a controlar-lo, el qual permet tasques bàsiques com ajustar la potencia de lectura, detectat antenes RFID o fins i tot escriure-hi.

Així doncs, el procediment que es seguia al laboratori per a realitzar la caracterització de les antenes dissenyades, consistia en una feina iterativa on el tècnic de laboratori havia d'ajustar manualment la potencia de lectura i comprovar quan l'antena era detectada (apropant o allunyat el tag) i repetint aquest procés per a cada posició. Com sembla evident aquesta era una feina molt repetitiva i que consumia una gran quantitat de temps perdut innecessàriament.

Es per això que des del departament es va decidir substituir aquest procés de mesura manual pel sistema automatitzat del que tracta aquest projecte.

2 Antecedents

Com acabem d'explicar, abans de que inicies aquest projecte, al departament d'enginyeria electrònica de la ESEIAAT ja utilitzaven el kit de desenvolupament M6e per a caracteritzar les antenes desenvolupades, però només d'una forma manual, i que va ser el fet de voler agilitzar el procés de mesura el que va potenciar l'inici d'aquest TFM.

No obstant, cal destacar que la idea d'utilitzar el kit de desenvolupament M6e dins d'un sistema automatitzat, no és original d'aquest projecte, doncs la idea de desenvolupar-lo va sorgir al departament de la universitat a arrel d'un projecte que ja van dur a terme a la universitat de Cambridge l'any 2016. [1]

Aquest projecte original, utilitzava el ja esmentat M6e juntament amb un motor a pas de 200 posicions per a caracteritzar les antenes RFID. El seu procés, mes complert que el descrit en aquest TFM, no nomes considerava els valors de la potencia de detecció, si no que també realitzava per a cada mesura una variació de la freqüència per a obtenir dades addicionals. A la figura 1 podem observar el sistema de mesura dissenyat l'any 2016:

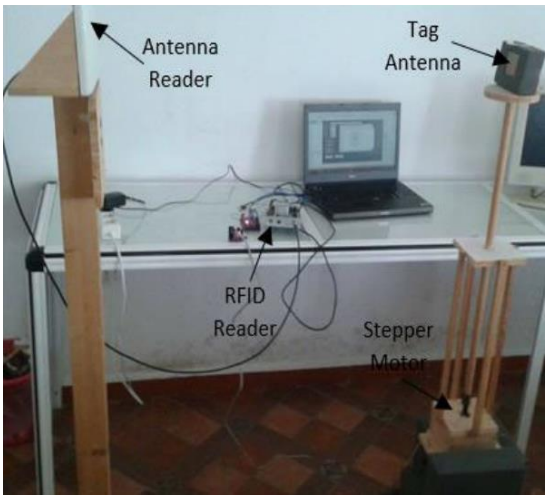

Figura 1. Sistema de mesura del projecte original (Font: Measurement system for over-the-air evaluation of UHF RFID tags quality. Wireless Power Transfer [1])

D'aquesta manera, l'objectiu d'aquest TFM no és el disseny des de zero d'una idea teòrica, sinó el procés d'enginyeria inversa per a dissenyar un nou sistema basat en el ja desenvolupat a Cambridge i que permeti obtenir els requeriments especificats prèviament.

Dintre d'aquest apartat d'antecedents, també voldria explicar per que vaig decidir seleccionar aquest TFM i la meva experiència prèvia pel que fa al disseny i a la programació de sistemes automatitzats.

De les assignatures cursades al llarg dels meus estudis universitaris, les que més m'han acabat agradant han sigut aquelles relacionades amb la programació i la automatització, fet que em va portar a realitzar ara farà 2 anys un treball de final de grau consistent en el disseny del software de control d'una maquina per a fer assaigs de curt circuit.

En aquell treball, on utilitzant el llenguatge C vaig haver de programar un microcontrolador, també vaig acabar tractant breument amb el programa Visual Studio per a modificar una aplicació ja existent. És per això pel que al trobar aquest projecte on, a part de poder desenvolupar les meves habilitats en l'àmbit de la programació aplicada, també podria acabar d'aprofundir en el programa Visual Studio, no vaig dubtar en acceptar-lo.

3 Fonament Teòric

3.1 Tecnologia RFID

La tecnologia d'Identificació per Radiofreqüència o RFID, basa el seu funcionament en els camps electromagnètics per a identificar els anomenats tags RFID, que es poden enganxar o incorporar a gairebé qualsevol objecte, proporcionant les dades assignades a aquests, així com la possibilitat de realitzar el seu seguiment. [2]

Un tag RFID és un element passiu que esta format principalment de 3 parts:

- Microchip: circuit integrat capaç d'emmagatzemar i processar informació
- Antena: encarregada de rebre i transmetre les senyals electromagnètiques
- Substrat: material que subjecta i manté units els elements del tag (pot ser tant paper, plàstic o teixit, sempre i quan no generi interferències)

A la figura 2 podem observar un dels tags dissenyats directament amb teixit als laboratoris del departament d'enginyeria electrònica de la ESEIAAT, on hi podem veure clarament els 3 elements presents: al centre de la imatge hi veiem el Microchip, "fil" gris, que es realitat un conductor, formaria l'antena i finalment el tros de teixit verd seria el substrat que ho manté tot unit.

Figura 2. Tag RFID dissenyat al laboratori ESEIAAT (Font: Fotografia pròpia)

Pel que fa al funcionament d'aquesta tecnologia, primerament cal entendre que els tags són uns element passius que no realitzen cap acció fins que no són alimentats per un camp electromagnètic generat per un emissor extern, el qual anomenem interrogador. Quan aquest interrogador emet el camp electromagnètic (amb una freqüència i potencia determinades), aquest és captat per l'antena del tag, la qual, actuant com a una bobina, genera un corrent elèctric que alimenta el circuit integrat del Microchip. Si la potencia de la senyal rebuda pel tag és suficient, el Microchip és capaç de transmetre a l'interrogador les dades que emmagatzemava, tancant així la comunicació.

Com podem observar, un dels elements clau per a que es produeixi l'intercanvi d'informació és el fet de que les antenes del tag puguin transformar el camp electromagnètic emès per l'interrogador en un corrent suficient per alimentar el Microchip i que aquest transmeti les seves dades. És de fet en aquest element on apareix la necessitat del procés de caracterització d'aquest treball.

La potencia que reben els tags del camp electromagnètic depèn principalment de tres elements: la distancia (contra més lluny es trobi el tag del interrogador més es debilitarà el camp), el guany o pèrdua derivats de la orientació del tag en referencia de l'emissor (les bobines que formen les antenes del tag no rebran la mateixa potencia del camp si la seva posició és perpendicular o paral·lela a aquest) i finalment, com és lògic, de la potencia inicial del camp generat per l'interrogador (contra més elevada sigui a la fon més elevada serà al receptor). Aquests tres elements apareixen a la anomenada *formula de Friis*, que permet determinar la potencia que rep una antena RFID, i amb la qual és pot realitzar la seva caracterització, procés que no deixa de ser una verificació del correcte funcionament d'aquestes antenes en diferents posicions.

Com hem vist prèviament, i en relació als tres factors anteriors, un procés utilitzat per a aquesta caracterització és la de fixar la potencia i modificar la distancia (apropant o allunyant el tag de l'interrogador) fins a que és produeixi l'intercanvi d'informació (l'interrogador detecti el tag), i repetir aquest procediment per a diferents posicions de l'antena.

Si bé aquest procés és senzill per assajos manuals, com els realitzats fins ara al laboratori de la ESEIAAT, per a un sistema automatitzat és més pràctic invertir el procés, fixant la distancia entre els dos elements i modificant la potencia del emissor, per a obtenir així les potencies de detecció.

Repetint aquest assaig per a les diferents posicions angulars al llarg d'un cercle s'obté el gràfic polar anomenat *diagrama de radiació* (figura 3), on es mostren aquestes potencies de detecció en funció de l'angle relatiu entre el tag i l'interrogador.

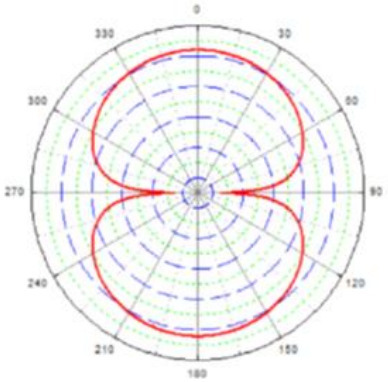

Figura 3. Diagrama de radiació tag RFID (Font: https://indjst.org/articles/design-of-a-miniaturized-dipole-rfid-tag-antenna)

Com podem observar, la forma habitual d'aquests diagrames de radiació és similar a un 8, això és deu a que quan el tag esta "mirant" directament al interrogador, és a dir esta en una posició paral·lela de 0 graus, o quan esta de "esquena", formant 180 graus, la detecció és màxima, mentre que quan la seva posició és perpendicular, a 90 i a 270 graus, la detecció és mínima.

3.2 Motors pas a pas

Com hem vist prèviament a l'abast del projecte, una gran part d'aquest consisteix en el control d'un motor pas a pas, però, que és exactament aquest element.

Un motor pas a pas o Stepper Motor, és un motor elèctric de corrent continua que, degut al seu disseny que li permet dividir la rotació en un nombre de passos iguals, té una gran precisió. [3]

Els motor de pas mes utilitzats son els bipolars, els quals, tal i com indica el seu nou, estan format per 2 bobines alimentades per 4 cables. Per a entendre el seu funcionament cal veure el seu interior, reflectit a la figura 4, on hi trobem principalment 2 elements: el rotor, consistent en una roda dentada imantada, i l'estator, el qual esta format per 4 parells de pols que tenen unes dents que nomes s'alineen amb les del rotor alternativament:

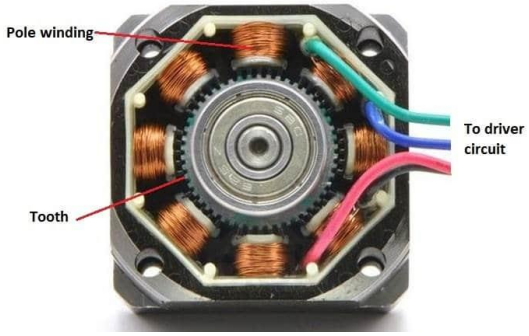

Figura 4. Interior d'un motor pas a pas (Font: https://www.homemade-circuits.com/stepper-motor-explained/)

Observant la figura anterior, podem veure com, al alimentar cada fase, generem una polaritat a cada una de les parelles de pols de l'estator, fet que provoca que les dents del rotor s'hi alineïn. D'aquesta manera, si intercanviem els pols alimentats en una seqüència concreta, generem el moviment de rotació. [4]

Una de les principals característiques que cal tenir en compte a l'hora de seleccionar un motor pas a pas és el numero de posicions que pot tenir dins d'una volta. Aquest valor ve donat pel numero de dents que té, ja que cada cop que intercanviem els cables a alimentar aconseguim moure una. D'aquesta manera, un motor a pas de 200 posicions (o passos), tindrà un rotor amb 200 dents, i cada cop que es realitzi un canvi de cables alimentat d'acord amb la seqüència adient, avançarà 1,8 graus (360/200).

Com hem vist un motor pas a pas bipolar compta amb 4 cables que formen 2 bobines, és per això que la nomenclatura que s'utilitza per a identificar-los és A+, A-, B+ i B-, sent A i B els identificadors de cada bobina. Tenint en compte aquesta regla, a la següent figura hi observem la seqüencia d'alimentació dels cables per a fer girar el motor en l'anomenat "Single Step":

Pas	A+	B+	A-	B-
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Figura 5. Seqüència "Single Step" (Font: Taula pròpia)

Addicionalment també existeix una seqüència anomenada "Double Step", la qual consisteix en alimentar dues fases simultàniament, i on el rotor s'alinea amb el punt intermig entre les dues fases alimentades. Un avantatge de fer servir aquesta seqüencia és el fet de que el motor realitza el doble de parell motor, pel que és una bona opció quan la carrega que ha de moure és elevada. No obstant, el fet d'utilitzar-la comporta haver de consumir el doble de potencia. A la figura 6 hi veiem l'ordre en que s'alimenten les fases amb aquest mètode:

Pas	A+	B+	A-	B-
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

Figura 6. Seqüència "Double Step" (Font: Taula pròpia)

Finalment, també existeix la possibilitat de doblar el nombre de posicions d'un motor a par utilitzant la seqüència anomenada "Half Step". En aquesta s'intercalen les posicions del "Single Step", on les dents del rotor s'alineen amb les del estator, i les del "Double Step", on les dents del rotor s'alineen entre 2 de l'estator. Si bé aconseguim el doble de posicions, el parell generat pel motor és força inferior, pel que no és una opció viable quan les carregues a moure son elevades. A la següent figura podem veure en que consisteix aquesta seqüència:

Pas	A+	B+	A-	B-
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

Figura 7. Seqüència "Half Step" (Font: Taula pròpia)

3.3 Visual Studio

El programa utilitzat per al disseny i programació del software de control així com la aplicació interactiva del sistema desenvolupat, va ser el Visual Studio.

Si bé en un inici es van tenir en consideració alternatives centrades en programes com Matlab, el qual oferia una gran cantat d'eines per a realitzar el tractament de les dades obtingudes als assajos, finalment es va decidir utilitzar el Visual principalment per dues raons:

La primera seria el fet de que, com veurem pròximament, l'empresa JADAK, desenvolupadora del kit de desenvolupament M6e, facilita a la seva pagina web tot tipus de programes d'exemple per a aprendre a programar el M6e amb els anomenats "Codelets", els quals centren el seu funcionament en el programa Visual Studio.

Si bé existia la possibilitat d'utilitzar aquest "Codelets" a altres programes, afegint-hi les llibreries pròpies, el segon factor que va acabar decantant la balança pel costat del Visual va ser la possibilitat que aquest oferia de desenvolupar-hi directament una aplicació amb la qual els usuaris poguessin interactuar i controlar el sistema.

A part, també va ser un factor important el fet de que durant el meu treball de final de grau hagués treballat breument amb el programa, i tenia moltes ganes de poder aprofundir-hi.

3.3.1 El programa

Microsoft Visual Studio és un entorn de desenvolupament integrat, o IDE, creat per Microsoft l'any 2002, que permet desenvolupar tant pagines web com programes informàtics amb interfícies gràfiques, entre altres. [5]

Figura 8. Logotip del programa Visual Studio (Font: https://es.wikipedia.org/wiki/Microsoft_Visual_Studio)

Visual Studio permet programar amb un total de 36 llenguatges informàtics diferents, entre es que destaquen C, C++, Visual Basic, .Net, HTML, Python o JavaScript. No obstant, de totes les opcions disponibles, la seleccionada per al desenvolupament d'aquest TFM va ser la de C# o C Sharp.

La selecció d'aquest llenguatge va ser provocada, novament, pel fet de que la gran majoria dels "Codelets" per a controlar el M6e estaven escrits en C#. Si bé aquest no era un llenguatge amb el que hagués treballat prèviament, el fet de tenir una amplia experiència en un altre de la seva família, el C++, tant per les assignatures cursades al llarg del meu pla d'estudis com per la feina feta al treball de final de grau, em va permetre aprendre'l amb relativa facilitat.

3.3.2 Aprenent Visual Studio

Com he explicat prèviament, la meva única experiència amb el programa Visual Studio va ser una petita modificació d'una aplicació ja existent per al treball de final de grau. És per això que, un cop presa la decisió d'utilitzar aquest programa per a la realització del projecte, vaig fer una formació autònoma principalment a base de vídeo-tutorials.

Dels diferents programes que vaig crear per a aprendre els secrets del Visual Studio, destacaré la creació d'una aplicació consistent en una calculadora. [6]

Al iniciar un nou projecte amb Visual Studio, la primera tasca a realitzar és la de especificar el tipus de programa que es vol crear. Com podem veure a la figura 9, cal seleccionar, a traves dels desplegables, el llenguatge de programació a utilitzar (en aquest cas va ser el **C#**), el sistema operatiu en el que ha de funcionar (**Windows** en el nostre cas) i el tipus de programa (per a aquest programa, al voler crear una app interactiva, es va seleccionar **Console**).

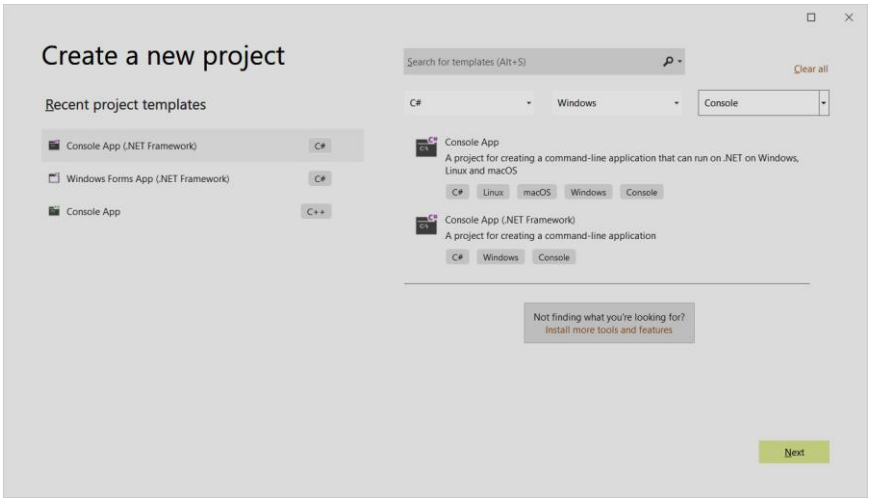

Figura 9. Pantalla crear projecte (Font: Captura Visual Studio)

Un cop creat el projecte, Visual Studio genera 2 espais de treball per a crear el programa: el de disseny de la aplicació i el de la programació del codi.

L'espai de disseny, és força intuïtiu, doncs consta principalment de les tres parts que es poden veure a la figura 10:

- Al lateral esquerra tenim un llistat amb els diferents elements afegibles a l'aplicació (botons, quadres de text, etiquetes...), pel que simplement cal arrossegant-los fins a la posició desitjada.

- Al centre tenim la pròpia aplicació, on es poden acabar d'ajustar les posicions dels elements afegits prèviament, eliminar-los o redimensionar la seva mida.

- Al lateral dret hi tenim la barra de propietats, amb la qual podem modificar tant el text dels elements afegits, com el seu color o el seu estat (per exemple podem deshabilitar botons o fer-los invisibles).

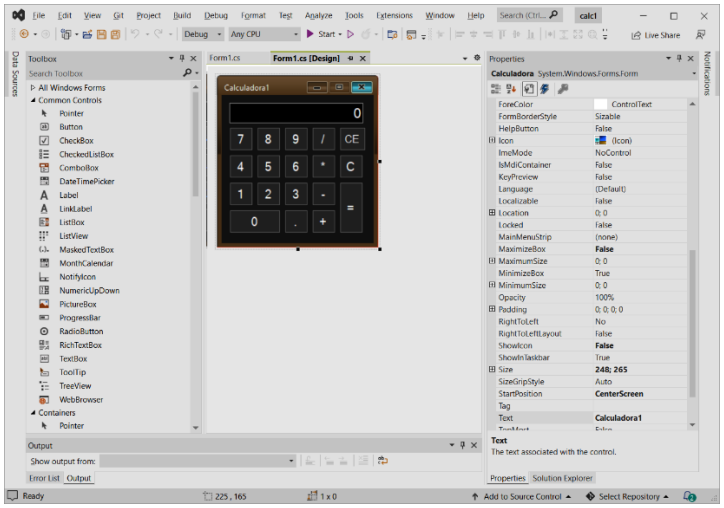

Figura 10. Espai disseny aplicació (Font: Captura Visual Studio)

Pel que fa a l'espai de programació, com podem veure a la figura 11, aquest és un compilador normal i corrent, amb l'únic tret característic de que requereix afegir ordres concretes per a cada element afegit a l'espai de disseny gràfic (per exemple si hem afegit un botó, caldrà crear el conjunt d'ordres a executar si aquest boto és clicat).

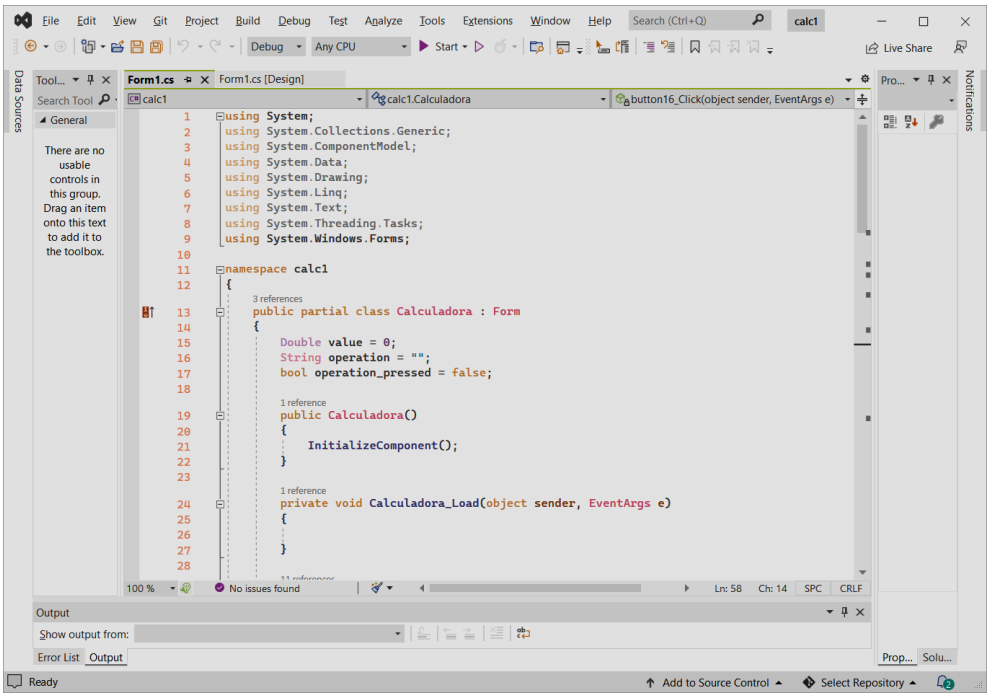

Figura 11. Espai de programació (Font: Captura Visual Studio)

Un cop finalitzada l'aplicació, Visual Studio no nomes permet executar-la a traves del programa, també genera una carpeta exportable (figura 12) amb l'executable del programa creat, pel que pot ser enviada a qualsevol altre ordinador per a que l'utilitzi sense la necessitat de que aquest tingui el Visual instal·lat.

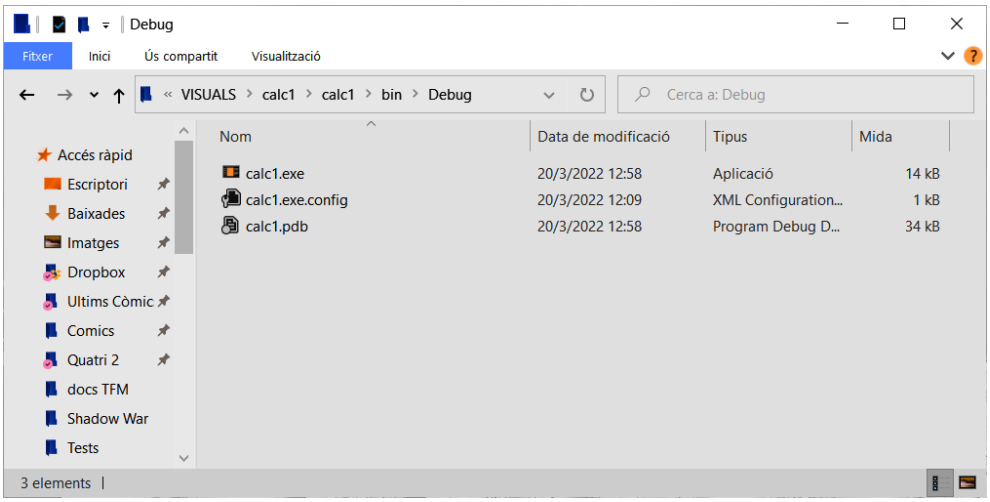

Figura 12. Carpeta amb programa executable (Font: Captura pròpia)

3.4 M6e

3.4.1 Kit de Desenvolupament

El principal element sobre el que gira aquest projecte és el ja anomenat kit de desenvolupament M6e el qual va ser desenvolupat per l'empresa americana ThingMagic Inc., que és una filial del grup empresarial d'innovació tecnològica JADAK i referent mundial en la fabricació i comercialització de productes de recopilació de dades des de l'any 2000, en que va ser fundada. [7]

Figura 13. Logotip empresa ThingMagic (Font: https://www.jadaktech.com/)

El kit M6e, el qual te un preu de 1370€, és una molt bona opció per a tot tipus d'aplicacions de la tecnologia RFID, doncs no només permet realitzar tasques simples com la de detecció i identificació d'objectes, sinó que també ofereix una amplia varietat de característiques addicionals com ara la capacitat d'escriure sobre tags RFID (per a emmagatzemar-hi dades per exemple) o la de poder ajustar fàcilment la potencia de lectura (element clau per al desenvolupament d'aquest projecte). [8]

Com podem veure a la figura 14, el contingut del paquet de desenvolupament, inclou primerament el mòdul integrat M6e, el qual, entre altres, compta amb fins a 4 sortides per antenes "interrogadores", connexió USB per a poder ser controlat per un ordinador, o 4 ports GPIO (general purpose input/output), essencials per al desenvolupament del sistema de posicionament. Dins del paquet de desenvolupament també s'inclou una antena "interrogadora", així com diversos tags RFID de mostra. Finalment, el "Devkit" també afegeix l'accés a diverses eines informàtiques, entre les que destaquen el programa Universal Reader Assistant o el paquet de dades Mercury API, el qual inclou entre altres els "Codelets", dels quals en parlarem a continuació.

Figura 14. Paquet de desenvolupament M6e (Font: https://www.jadaktech.com/products/thingmagic-rfid/thingmagic-uhf-rfid-modules-development-kit/)

Com explicàvem als antecedents d'aquest projecte, abans del seu inici, al departament d'enginyeria elèctrica de la ESEIAAT, ja utilitzaven el M6e per a la caracterització d'antenes RFID d'una manera manual, procés es realitzava amb ja esmentat Universal Reader Assistant, que podem veure a la figura 15. Aquesta aplicació, propietària de ThingMagic,

permet realitzar tot tipus d'assajos amb el M6e, entre els que destaquen els ja esmentats per a escriure sobre els tags, o el que permeten detectar tags a diferents potencies. Per a aquest darrer procés, el problema que tenia aquesta aplicació, era el fet de que requeria introduir manualment la potencia de lectura cada cop que es volia canviar aquesta, i era necessari aturar i arrancar el sistema. Degut a això, per a realitzar a caracterització completa d'una antena RFID els temps necessaris eren excessius, pel que es va decidir la incorporació del sistema del que tracta aquest projecte.

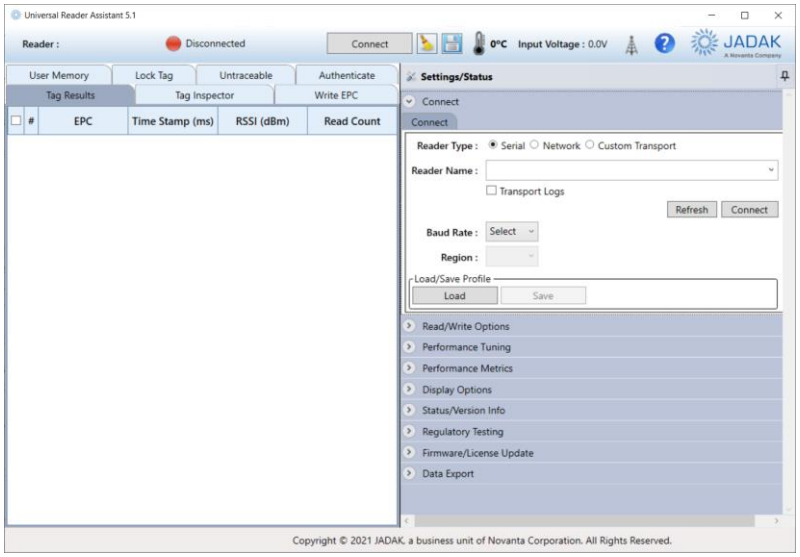

Figura 15. Programa Universal Reader Assistant (Font: Captura del programa)

Com hem esmentat, el programa assistent de lectura no és l'únic programari inclòs en el kit de desenvolupament, també compta amb el paquet de dades Mercury API. Aquest paquet, consistent en una carpeta comprimida en format ZIP, conté tot tipus d'eines per a la programació del M6e, entre les que es compten diversos programes d'exemple executable i els anomenats "Codelets", codis de mostra per a diverses funcions del M6e, com per exemple el "Read", que activa l'antena emissora durant 1 segon i retorna una llista amb els tags detectats, "ReaderStats", que permet modificar i comprovar les propietats de les lectures (potencia, freqüència, regió, antena utilitzada...) o el "GPIOCommands", que permet controlar els ports GPIO. Aquests "Codelets", en llenguatge C# per a Visual Studio, a part del propi codi i la seva explicació, també inclouen les llibreries necessàries per a poder compilar i executar els programes, pel que es van convertir en una eina essencial per al desenvolupament d'aquest TFM.

A part dels elements inclosos en el Mercury API, també em van ser de gran ajut a l'hora de programar i controlar el M6e el servei d'atenció client que JEDAK ofereix, el qual em va ajudar a resoldre alguns problemes puntuals, així com el seu manual de programació, al qual vaig haver de recórrer per a identificar algunes funcions que no estaven especificades al "Codelets".

Com hem explicat prèviament, les tasques que ha de realitzar el M6e en el sistema desenvolupat són bàsicament 2: ha de ser capaç de mesurar la potencia de detecció dels tags RFID, i, a part, ha de poder controlar un motor pas a pas amb els seus ports GPIO (que en aquest cas treballaran com a sortida digital). En els següents apartats realitzarem l'estudi dels "Codelets" per a identificar les ordres necessàries per a realitzar cada una d'aquestes accions, i així poder-les incorporar al software de control del sistema desenvolupat.

3.4.2 Procés de Mesura

Com hem vist prèviament, el procediment per a realitzar una mesura consisteix en ajustar progressivament la potencia de lectura de l'antena emissora (interrogador) fins a que aquesta sigui suficient per a detectar els tags RFID que estem caracteritzant. Per tant, hem d'aconseguir identificar per una banda les ordres que permetin ajustar les potencies de lectura i per una altra les que permetin comprovar si es detecta el tag o no.

3.4.2.1 Ajustar potència

Una de les primeres activitats que vaig dur a terme amb el M6e, va ser la de realitzar un tutorial sobre com crear nou programes a partir dels codelets. Aquest primer programa, que era força senzill, a partir del "Codelet" "Reader.Information".

Aquest programa, primerament intentava connectar amb el M6e, procés per al qual requeria la introducció del port de comunicació (aquest element es utilitzat per l'ordinador de forma interna per a identificar el port USB utilitzat per a connectar amb el M6e). Com podem veure a la figura 16, abans de poder realitzar la connexió amb el M6e, cal crear un objecte "Reader", el qual requereix del ja explicat port de comunicació en el format ***tmr:///COM3*** (en cas de que el port de comunicació sigui el COM3) i que en el "Codelet" esta emmagatzemat a la variable args[0]. Un cop creat aquest objecte, ja es pot realitzar la connexió amb l'ordre **r.Connect().**

```
using (Reader r = Reader.Create(args[0]))
{
    r.Connect();
```

Figura 16. Connectar amb el M6e (Font: Captura del "Codelet")

El programa continuava amb un conjunt d'ordres on simplement es generava un llistat les característiques del M6e (model, numero de sèrie, versió del software...), fins a un conjunt d'ordres que permetien identificar i modificar la potencia de lectura. Com podem observar a la figura 17, amb l'ordre ***r.ParamSet("/reader/radio/readPower", power)*** podem ajustar la potencia al valor emmagatzemat a la variable ***power***, potencia que, d'acord amb el manual del M6e, en dBm per al M6e pot tenir valors entre els 5 dBm i els 31 dBm amb augments amb augments de 0,5 dBm. [9]

```
//  Method to set the value RF read power
power = 2000; // 20 dBm
r.ParamSet("/reader/radio/readPower", power);
power = Convert.ToInt32(r.ParamGet("/reader/radio/readPower"));
Console.WriteLine("Global ReadPower is set to: " + power);
Console.WriteLine("********** Step-1 Completed ********** ");
```

Figura 17. Ajustar potencia lectura M6e (Font: Captura del "Codelet")

3.4.2.2 Detectar tags RFID

Si bé no existeix cap "Codelet" que permeti comprovar si es detecta algun tag, el programa "Read" activa l'antena emissora durant un període de temps especificat per a posteriorment generar una llista amb les tags detectats. A partir d'aquest procediment, simplement comprovant si la llista amb els tags identificats és buida o no, podem verificar si el tag ha estat detectat o no.

Aquest "Codelet", al igual que l'utilitzat a l'apartat anterior, inicia creant un objecte "Reader" i connectant-se amb el M6e. Com podem observar a la figura 18, un cop establerta la connexió, el primer que fa el programa es comprovar si la regió seleccionada es compatible amb el M6e per a posteriorment ajustar-la amb l'ordre ***r.ParamSet("/reader/region/id",...)***.

```
r.Connect();
if (Reader.Region.UNSPEC == (Reader.Region)r.ParamGet("/reader/region/id"))
{
    Reader.Region[] supportedRegions = (Reader.Region[])r.ParamGet("/reader/region/supportedRegions");
    if (supportedRegions.Length < 1)
    {
        throw new FAULT_INVALID_REGION_Exception();
    }
    r.ParamSet("/reader/region/id", supportedRegions[0]);
}
```

Figura 18. Ajustar regió M6e (Font: Captura del "Codelet")

Seguidament el programa verifica si l'antena seleccionada per a l'assaig esta activada, amb l'ordre ***r.isAntDetectEnabled(antennaList)*** present a la figura 19.

```
if (r.isAntDetectEnabled(antennaList))
{
    Console.WriteLine("Module doesn't has antenna detection support please provide antenna list");
    Usage();
}
```

Figura 19. Comprovar antena M6e (Font: Captura del "Codelet")

Finalment, un cop comprovades i ajustades les variables de l'assaig, el programa procedeix amb la mesura (figura 20), primer creant un "pla" de lectura on hi introdueix les dades prèvies, i posteriorment realitzant la mesura amb l'ordre ***TagReadData[] tagReads = r.Read(500)*** (on el 500 representa la durada de la lectura en milisegons) que emmagatzema el llistat dels tags identificats a la variable ***tagReads***, que serà la que comprovarem si es buida per a verificar si els tags han estat detectats o no.

```
// Create a simplereadplan which uses the antenna list created above
SimpleReadPlan plan;
plan = new SimpleReadPlan(antennaList, TagProtocol.GEN2, null, null, 1000);

// Set the created readplan
r.ParamSet("/reader/read/plan", plan);

// Read tags
TagReadData[] tagReads = r.Read(500);
```

Figura 20. Realitzar lectura M6e (Font: Captura del "Codelet")

3.4.3 Control ports GPIO

Com hem vist prèviament el "Codelet" "GPIOCommands", inclou les ordres necessàries per a controlar els ports GPIO del M6e, no obstant, aquest va ser un dels apartats que va generar més problemes.

D'entrada, analitzant el programa d'exemple, el qual, com ja es habitual, s'inicia creant l'objecte "Reader" i connectant amb el M6e, son fàcilment identificables les ordres per a ajustar el valor dels ports de sortida (o outputs), els quals, com veiem a la figura 21, requereixen d'una variable ***gps*** que es generada pel subprograma ***ArrayListToGpioPinArray***, i que executa l'ordre ***r.GpoSet(gps)***.

```
GpioPin[] gps = ArrayListToGpioPinArray(list);
r.GpoSet(gps);
```

Figura 21. Ajustar ports GPIO (Font: Captura del "Codelet")

Per a comprendre aquesta darrera operació, primerament cal analitzar exactament que fa el subprograma i quin es el format exacte de la variable ***gps*** que conte la informació de quin ports han d'estar activats i quins desactivats. Si bé d'entrada pot semblar un conjunt d'ordres força complexes, en realitat nomes ens son veritablement útils les darreres ordres. Amb l'expressió ***GpioPin gp = new GpioPin(id, high)*** simplement estem assignant el valor de cada port, sent ***id*** el numero del port (1, 2, 3 o 4) i sent high un booleà que pren valor 1 qual el port ha de estar activat i 0 quan aquest ha d'estar descativat. A part, l'expressió ***gps.Add(gp)*** simplement agrupa els diferents ***gp*** de cada port en el conjunt ***gps*** que els engloba a tots.

```
private static GpioPin[] ArrayListToGpioPinArray(ArrayList list)
{
    List<GpioPin> gps = new List<GpioPin>();
    foreach (ArrayList innerlist in list)
    {
        int[] pinval = (int[])innerlist.ToArray(typeof(int));
        int id = pinval[0];
        bool high = (0 != pinval[1]);
        GpioPin gp = new GpioPin(id, high);
        gps.Add(gp);
    }
    return gps.ToArray();
}
```

Figura 22. Subprograma variable gps (Font: Captura del "Codelet")

Si bé podríem pensar que amb aquestes ordes ja seriem capaços d'activar i desactivar els ports GPIO a voluntat, per experiència pròpia, al executar únicament aquestes ordres el M6e retorna un error indicant el següent "*Error: The reader received a valid command with an unsupported or invalid parameter*".

Després de contactar amb el ja esmentat suport tècnic de JEDAK, em van indicar que per a poder ajustar el valor d'un port GPIO amb la comanda ***r.GpoSet(gps)***, primerament calia ajustar el port en qüestió com a una sortida digital, tot utilitzant l'ordre

r.ParamSet("/reader/gpio/outputList", output), on la variable ***outputList*** simplement es un llistat amb els ports que es volen assignar com a sortides digitals o outputs.

Un cop recopilat tot això, vaig ser capaç de dissenyar un primer programa amb el Visual Studio per a poder controlar els ports GPIO, el qual, com podem veure a la figura 23, compta bàsicament de 3 botons i d'un quadre de text. Per a ajustar el ports, primerament l'usuari s'ha de connectar amb el M6e amb el botó "CONECT", seguidament, en el quadre de text introdueix els estats dels ports en el format indicat (en aquest cast tots 4 ports estarien desactivats al tenir un 0). Un cop introduïdes les dades, amb el botó "SET" s'ajustarien el valors de cada port, i finalment amb el "STOP" es desconnectarien tots els ports per a deixar-lo en l'estat de repòs.

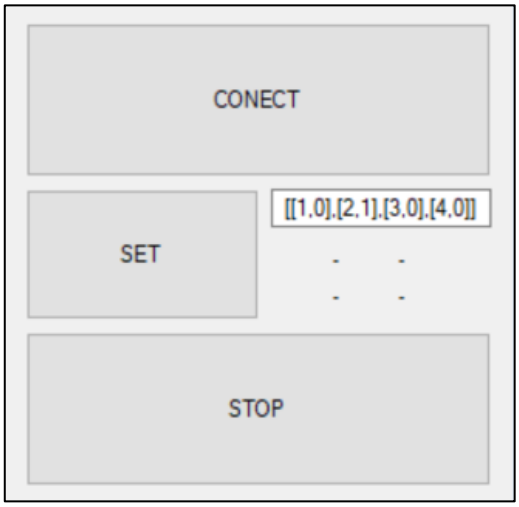

Figura 23. Programa ajustar ports GPIO (Font: Captura pròpia)

Per a comprovar el correcte funcionament d'aquest programa, i abans de tenir el sistema de posicionament muntat, pel que no ho podia verificar amb el moviment del motor, vaig utilitzar un conjunt de 4 LEDs continguts al mòdul integrat del M6e (a la part superior esquerra de la figura 24) els quals podien ser activats i desactivats per les pròpies senyals dels ports GPIO a traves d'uns Jumpers.

Figura 24. LEDs mòdul M6e (Font: Fotografia pròpia)

Finalment, i per a verificar la possibilitat d'activar i desactivar els ports GPIO seguin la seqüència explicada al apartat 3.2 sobre els motors de pas, també vaig crear el programa present a la figura 25, el qual no nomes permetia activar el conjunt de ports d'acord amb cada posició (amb els botons Posició 1, 2, 3 i 4), sinó que també oferia la possibilitat de realitzar un bucle infinit (nomes aturable pel botó "STOP") on els ports s'activaven i desactivaven seguint la seqüència desitjada.

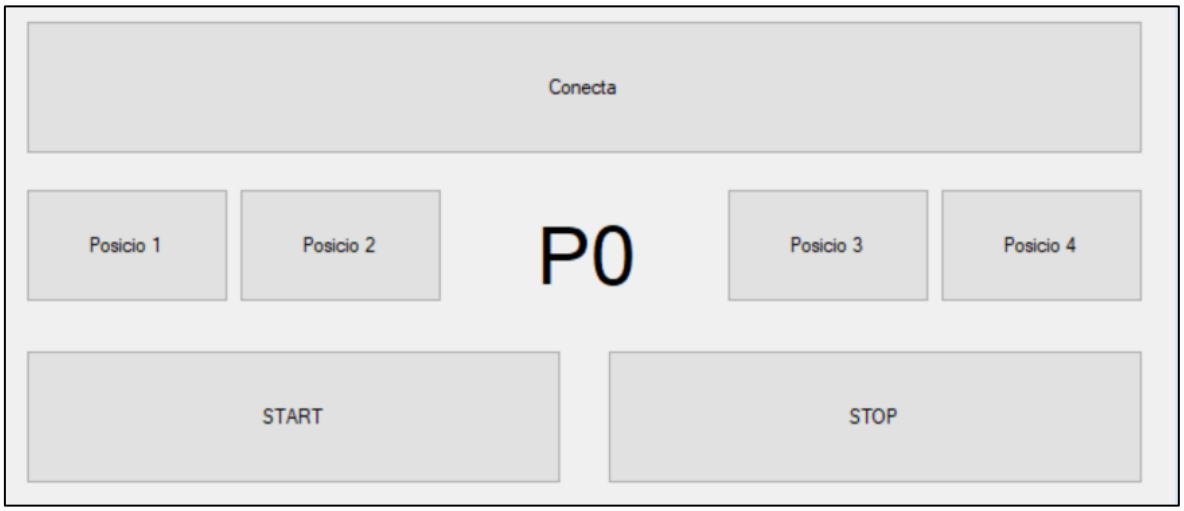

Figura 25. Segon programa ajustar ports GPIO (Font: Captura pròpia)

4 Desenvolupament

4.1 Sistema de posicionament

4.1.1 Selecció dels components

Com hem vist prèviament, un dels objectius d'aquest TFM és el d'aconseguir controlar un motor a pas amb el M6e. Per a fer-ho, ens utilitzarem dels seus ports GPIO (General Purpose Input/Output), que rebem aquest nom pel fet de que poden operar tant com a entrades digitals com a sortides. D'acord amb el manual del Hardware del M6e, aquest ports, quan treballen com a outputs, generen senyals de 3,3 V i uns 0,3 mA. Com sembla evident, amb aquestes senyals tant dèbils no es possible alimentar el motor directament, pel que a part del motor, també s'haurà d'adquirir un driver (o controlador) que sigui capaç d'ajustar el Stepper a partir de les 4 senyals digitals. [10]

Per a seleccionar aquests components, el director del TFM va recomanar buscar-los a les tendes digitals de "RS Components" i a la de "Farnell", empreses que tenen conveni amb l'escola i que permetien el subministrament dels productes en pocs dies.

En tant que el driver a seleccionar dependrà de les característiques del motor, primerament haurem d'identificar aquest element. Per a fer-ho, cal comprendre les dues principals característiques que defineixen un motor a pas: el parell motor i el seu pas.

Amb parell motor, ens referim a la "força" que es capaç de generar aquest, en unitats de Nm, es a dir, si enganxéssim una barra perpendicularment a l'eix del motor, quina seria la força que podríem generar en funció de la distancia fins a l'eix. Per a aquest projecte, el parell motor no requereix de ser especialment elevat, en tant que, com veurem més endavant, els únics elements que aquest ha de moure són força lleugers.

El pas del motor, com hem vist prèviament, representa el numero de posicions que aquet pot prendre dintre d'una volta, el qual equival al nombre de dents que te el cilindre del rotor. En tant que en el projecte original realitzat a Cambridge, van utilitzar un motor amb pas de 1,8º (200 posicions per volta), es va decidir utilitzar aquest per al nostre projecte.

Un cop identificades aquestes característiques tècniques, podem procedir a cercar un model que s'hi adapti. Com podem veure a la figura 26, la botiga digital de RS, compta amb una secció dedicada als motors pas a pas, la qual permet filtrar els diferents models en funció de les seves característiques:

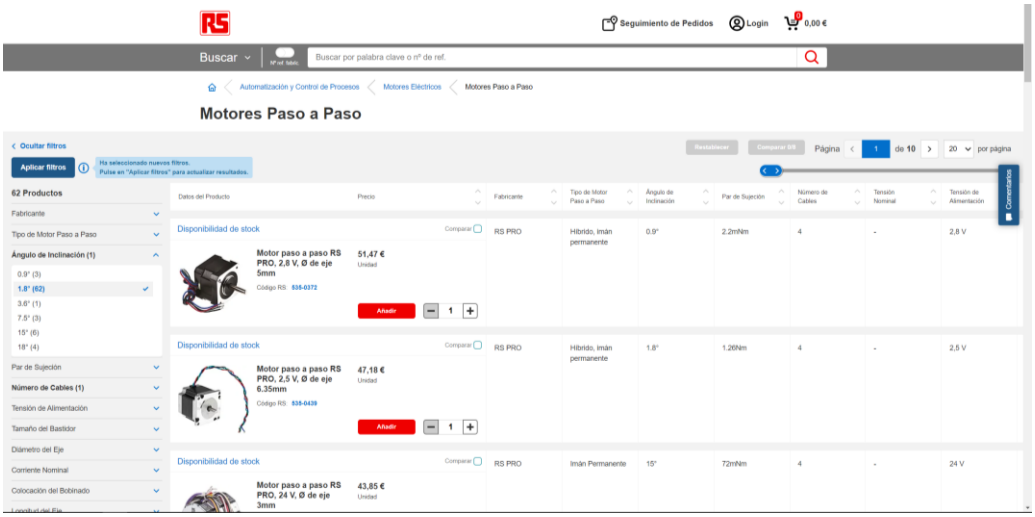

Figura 26. Motors a pas botiga RS (Font: https://es.rs-online.com/web/c/automatizacion-y-control-de-procesos/motores-electricos/motores-paso-a-paso/)

Dels diferents motors que complien amb els requisits explicats, hi destaquen els següents 2 models:

- Motor pas a pas RS PRO 535-0439. Aquest motor a pas bipolar, amb un parell de 1,26 Nm i pas de 1,8º, requereix de 2,5 V i 2,8 A per a la seva alimentació, i té un preu de 57,09 €. [11]

Figura 27. Motor RS PRO (535-0439) (Font: https://es.rs-online.com/web/p/motores-paso-a-paso/5350439)

- Motor pas a pas RS PRO 180-5283. Aquest motor a pas bipolar, amb un parell de 0,497 Nm i pas de 1,8º, requereix de 12 V i 0,42 A per a la seva alimentació, i té un preu de 58,08 €. [12]

Figura 28. Motor RS PRO (180-5283) (Font: https://es.rs-online.com/web/p/motores-paso-a-paso/1805283)

Com podem veure, tots dos tenen preus força similars i un mateix pas, però divergeixen bastant pel que fa al seu parell i a la seva alimentació. En tant que, com hem dit prèviament, l'estructura a moure no te un pes excessiu, no tindrem en compte el parell motor com a element decisiu, pel que serà en funció dels drivers que identifiquem a continuació, que podrem seleccionar el motor que millor s'adapti a aquests a partir de l'alimentació.

La selecció del driver, va ser una altra de les tasques on van sorgir una gran quantitat de problemes. Primerament, ni al web de RS ni al de Farnell, vaig ser capaç d'identificar cap driver que s'ajustés a cap dels motors explicats prèviament, pel que vaig haver de contactar amb el servei tècnic de RS per a que ells mateixos em recomanessin drivers adients per als seus motors. D'aquesta manera em van facilitar les referencies d'aquests 2 elements:

- Controlador de motor DC trifàsic Faulhaber. Aquest driver, que en realitat esta dissenyat per a motors trifàsics, pot ser reprogramat per a treballar com a driver de motors a pas. Amb un voltatge que pot ser ajustat entre els 0 fins als 28 V de continua, i amb un

amperatge de 8 A, és una bona opció per als 2 motors, si bé cal tenir en compte que té un preu de 401,93€. [13]

Figura 29. Controlador de motor DC trifàsic Faulhaber (Font: https://es.rs-online.com/web/p/controladores-de-motores/9211331)

- Controlador de motor pas a pas RS PRO 905-8786. Aquest driver de motors a pas, que pot treballar amb voltatges entre els 24 i els 48 volts i amb un amperatge de 2,2A, és a priori, una opció que no semblaria adient per a cap dels dos motors degut al fet de que te un voltatge molt superior al nominal d'aquests. No obstant, des de RS, van indicar que per al segon motor podria ser una opció viable. Cal destacar que el preu d'aquest segon driver és força inferior al del primer, sent aquest de 144,03€. [14]

Figura 30. Controlador de motor pas a pas RS PRO 905-8786 (Font: https://es.rs-online.com/web/p/controladores-de-motores/9058786)

A partir d'aquesta recerca al web de RS, semblava que la millor opció, sobretot per ser la mes econòmica, seria la de seleccionar el motor RS PRO 180-5283 (el de 12 volts amb un parell inferior) amb el driver RS PRO 905-8786 (més econòmic amb un voltatge entre els 24 i els 48 V). No obstant, el fet d'haver d'utilitzar un driver amb un voltatge nominal doble que el del motor, a part del fet de que tot i ser mes econòmica, aquest opció tenia un cost de gairebé 200 €, va potenciar la recerca d'alternatives fora de les botigues relacionades amb la UPC.

Com hem dit prèviament, els ports GPIO del M6e generen senyals de 3V, valor molt similar a les generades per les plaques Arduino utilitzades en tot tipus de sistemes automatitzats. Es per això que es va iniciar una recerca de projectes que utilitzessin aquest mòdul de programació per a controlar motors a pas, la qual va concloure amb la trobada del projecte "*Control Stepper Motor with L298N Motor Driver & Arduino*". [15]

Aquest disseny basa el seu funcionament en l'ús d'un pont en H, que és un circuit electrònic capaç d'invertir la polaritat del corrent aplicat a una carrega. Com podem veure a la figura 31, el funcionament d'aquest element és relativament senzill, consta de 4 interruptors

aparellats diagonalment (el superior esquerra amb l'inferior dret i l'inferior esquerra amb el superior dret) que permeten alimentar els borns positius d'un motor o els negatius. [16]

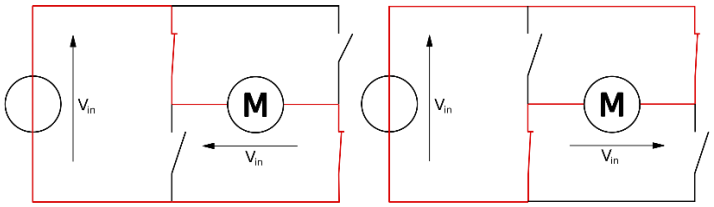

Figura 31. Pont en H (Font: https://en.wikipedia.org/wiki/H-bridge)

En el projecte identificat, s'utilitzava el mòdul integrat L298N, el qual esta format per 2 ponts en H connectats al motor tal i com veiem a la figura 32. La base del funcionament es que, polaritzant positiva o negativament cada bobina del motor a pas, es pot aplicar la seqüencia que ja havíem vist en l'apartat 3.2.

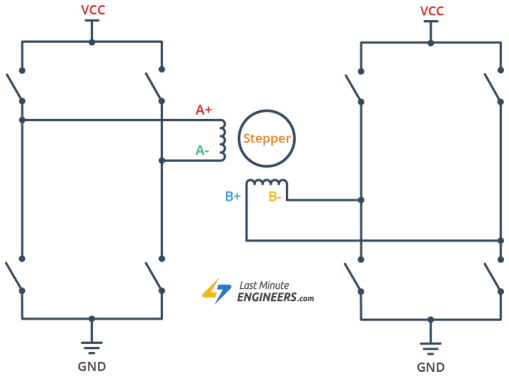

Figura 32. L298 projecte Arduino (Font: https://lastminuteengineers.com/stepper-motor-l298n-arduino-tutorial/)

Per a controlar els ponts en H, el L298N compta amb 4 entrades digitals de 5V (tot i que els ports GPIO del M6e només envien 3V, el L298N té una certa tolerància que és capaç de detectar les senyals inferiors), les qual en el projecte d'Arduino s'utilitzaven tal i com es veu a la figura 33.

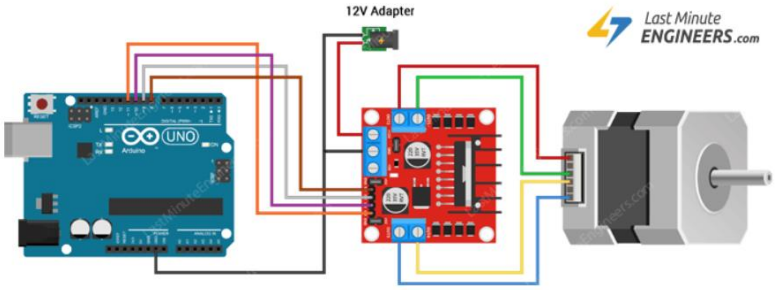

Figura 33. Diagrama elèctric projecte Arduino (Font: https://lastminuteengineers.com/stepper-motor-l298n-arduino-tutorial/)

D'aquesta opció, on a part del ja esmentat L298N, s'utilitzava un motor NEMA 17 de 12 volts i amb un pas 1,8º (200 posicions), n'és especialment destacable el cost total, doncs

el preu del driver és de tan sols 5,90€ i el del motor de 16,99 €, motiu pel que es va decidir utilitzar aquest mateixos components per a l'elaboració del projecte. [17] [18]

Cal destacar, no obstant, que la selecció d'aquests elements més econòmics, va generar algunes complicacions a l'hora d'implementar el sistema de posicionament.

Per una banda, el motor Longruner Nema 17 (figura 34) té un parell de 0,45 Nm, lleugerament inferior al dels motors identificats a RS, el qual va provocar a l'hora de realitzar proves al laboratori, "s'encallés" de tant en tant, es a dir, que no era capaç de generar un parell suficient per a moure l'estructura. No obstant, es va aconseguir solucionar aquest problema fent ús del ja explicat "Double Step", seqüencia on s'alimenten dues bobines a cada pas per a generar un parell el doble de fort.

Figura 34. Motor Longruner Nema 17 (Font: https://www.amazon.es/Longruner-Impresora-4-Cables-Conector-LD08/dp/B07FKH52S5?th=1)

A part, pel que refereix al driver L298N (figura 35), es va detectar que patia d'un problema de sobreescalfament quan havia de mantenir diverses bobines alimentades, problema que provocava que aleatòriament deixes de funcionar momentàniament. Per a solventar aquest problema, i com veurem quan analitzem en profunditat el codi del software de control, es va decidir afegir uns temps de "repòs" entre mesures, per a facilitar el refredament del driver, així com fer que el motor només fos alimentat a l'hora de girar-lo (en els codis originals, un cop ajustada la posició del motor aquest es mantenia alimentat per a "enclavar" la seva posició, no obstant, en tant que el motor esta en una superfície llisa i no rep cap carregar que el forci a girar, aquest enclavament en realitat és innecessari).

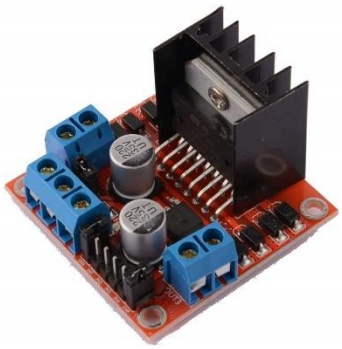

Figura 35. Driver L298N (Font: https://www.amazon.es/Movilideas-Puente-Bridge-Stepper-Controlador/dp/B089DPPKQ1?th=1)

4.1.2 Muntatge elèctric

Un cop seleccionats els components, la següent tasca va ser la de dissenyar i muntar el sistema elèctric per al posicionament de les antenes. Com podem veure a la figura 36, aquest esta basant en el del projecte d'Arduino de l'apartat anterior, on s'ha substituït aquest mòdul de programació pel M6e.

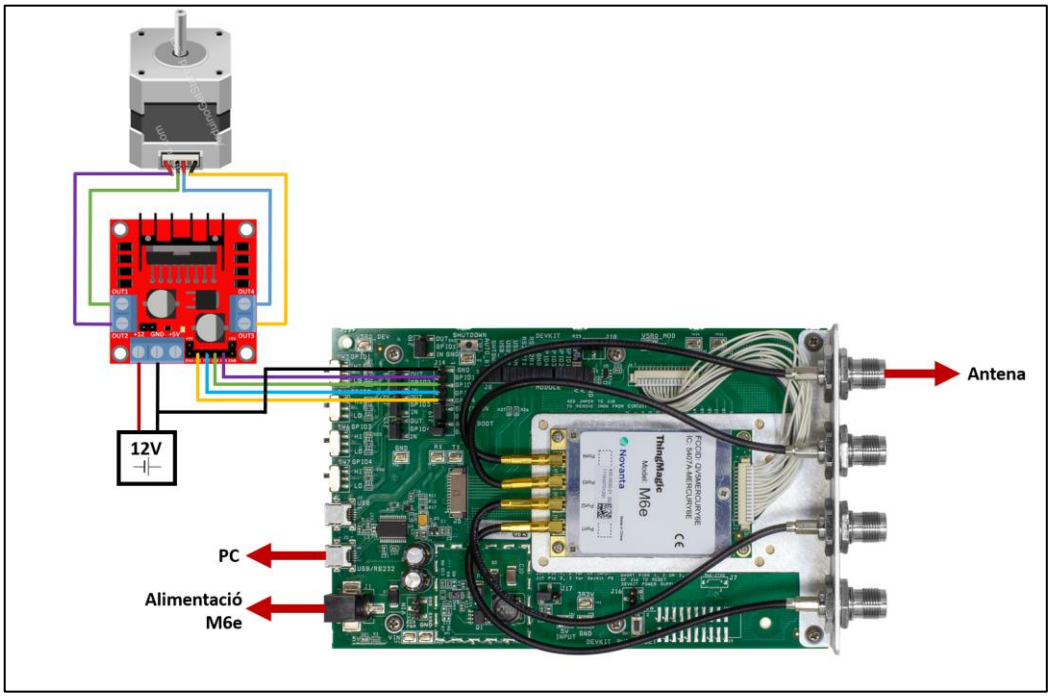

Figura 36. Esquema elèctric (Font: Diagrama propi)

Com es pot apreciar, del M6e surten, d'una banda, 4 cables amb les senyals de control generades pels ports GPIO que van al L298N. A part, i per a que les senyals puguin ser identificades, també hi surt un cable (en color negre al diagrama) el qual uneix el "GOUND" del M6e amb el del driver i el de la font de 12 V.

La font de 12V que alimenta el driver L298N, és la RS Pro SPD3303C (figura 37), la qual ja era present al laboratori, pel que no va ser necessària l'adquisició d'una nova. Aquesta font regulable permet generar corrents continues de fins a 32 V amb una intensitat màxima de 3,2 A, la qual és suficient per al sistema de posicionament dissenyat en aquest projecte.

Figura 37. Font d'alimentació RS Pro SPD3303C (Font: Fotografia pròpia)

Pel que fa a les connexions entre el motor i el L298N, va ser necessari identificar les bobines del motor, doncs d'aquest només hi sortien 4 cables sense identificar. Amb l'ús d'un multímetre va ser fàcil relacionar el cables que formaven cada una de les dues bobines, en tant que tancaven un circuit tancat, pel que un cop identificats només va caldre connectar cada parella a un dels 2 ponts en H del driver.

Addicionalment en el esquema també s´hi poden apreciar els diferents cables que surten del M6e, els quals són l'alimentació del propi M6e (que va a un endoll de paret), un cable USB que va al ordinador amb el que es controla el sistema, i un cable cap a l'antena emissora (interrogadora).

A la figura 38 podem veure el muntatge real dels sistema descrit. Com es pot apreciar, es va utilitzar una placa de connexions (anomenada "protoboard") per a els cables que porten les senyals des dels ports GPIO del M6e fins al driver del motor. Si bé per aquest es un muntatge temporal, de cara a al futur es preveu substituir aquesta placa de probes per un circuit soldat, el qual fixi millor unes connexions que ara per ara són fàcilment modificables.

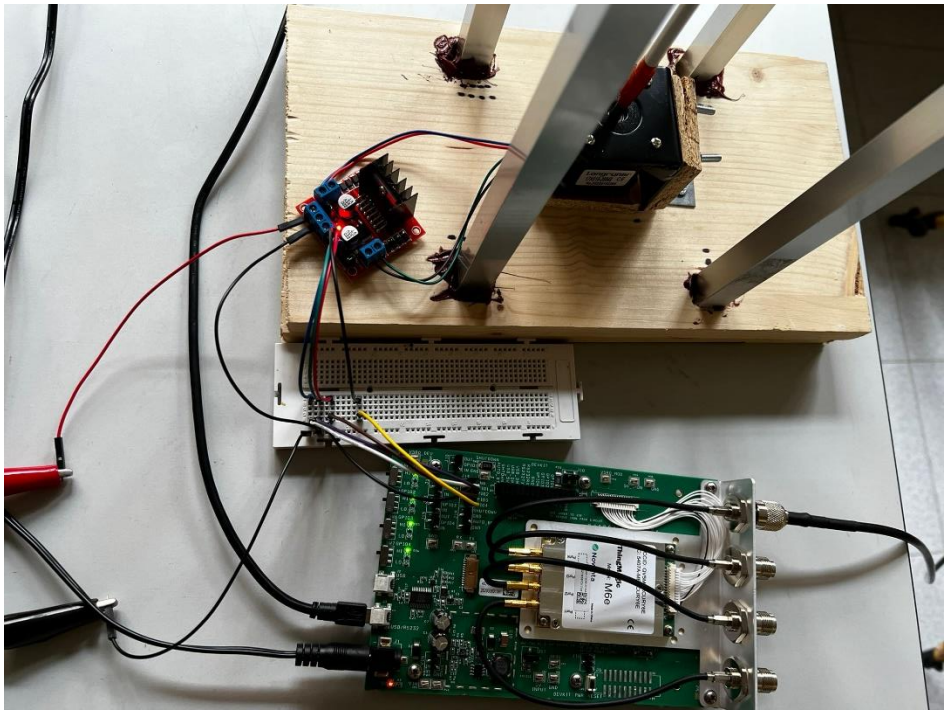

Figura 38. Muntatge real (Font: Fotografia pròpia)

4.1.3 Estructura mecànica

Si bé, com acabem de veure, el sistema elèctric ja esta dissenyat, encara falta una estructura mecànica que sigui capaç de traslladar el gir del motor als diferents tags RFID que volem caracteritzat.

Per al disseny d'aquesta estructura es van observar diferents possibilitats, com la de la figura 39, un primer disseny realitzat amb Solid Works i on el moviment de l'eix es traslladava a una "tarima" on col·locar-hi les antenes amb una pinça de plastic (el requadre de color gris) present al laboratori.

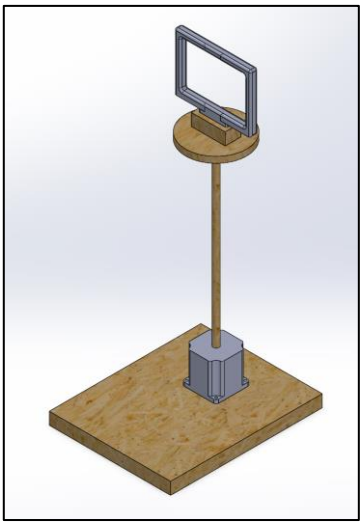

Figura 39. Primer disseny estructura (Font: Captura Solid Works)

Aquest primer disseny va ser desestimat principalment pel fet de que incloure tants elements sobre l'eix, augmentaria el parell que aquest hauria de generar, el qual podria no ser suficient. A part, com sembla evident, el fet d'elevar un eix sense cap tipus de suport podria provocar inestabilitats en l'estructura i fins i tot bolcaments. Es per això que després de diverses iteracions es va acabar amb el disseny de la figura 40.

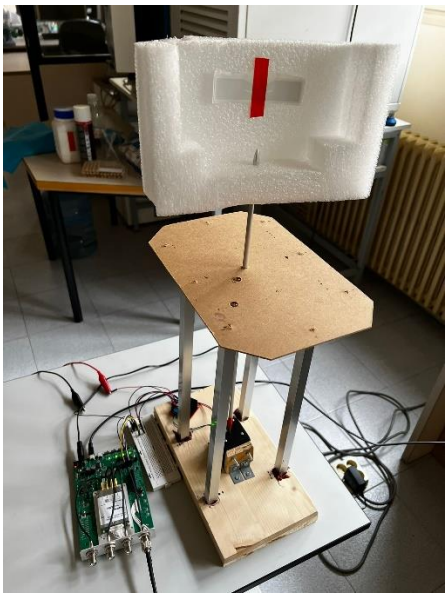

Figura 40. Estructura real (Font: Fotografia pròpia)

Com es pot observar, aquesta estructura esta formada, primerament, per una base de fusta, on esta enclavat el motor, i d'on i surten 4 potes d'alumini que subjecten una "tarima" de fusta que aporta estabilitat a l'eix de gir. Aquest eix de gir, unit al del motor amb cinta adhesiva (taronja), també esta fet d'alumini, i culmina amb una punta punxeguda la qual enclava una peça de poliestirè on es col·loquen els tags a mesurar (en el cas de la imatge el tag hi esta enganxat també amb cinta adhesiva taronja).

La principal avantatge d'aquest model es el fet de que les úniques peces que ha de moure el motor, i per tant el requeriment de parell, són la barra d'alumini que treballa com a eix de gir i la peça de poliestirè, la qual es extremadament lleugera. Es de fet per aquesta propietat que es va decidir utilitzar-la, a part del fet de que el poliestirè es un material que no genera interferències electromagnètiques, pel que fins i tot quan el tag esta "d'esquenes" de l'emissor (i per tant té el material entre mig) no disminueix la potencia rebuda.

Cal destacar que la fabricació de l'estructura va ser feta a partir d'elements reciclats 100% dels quals ja disposava, pel que es pot considerar com a negligible el seu cost.

Amb el sistema elèctric i l'estructura mecànica dissenyades, finalment podem observar a la figura 41 el sistema de posicionament real, on a la part esquerra hi tindríem tant el M6e com l'estructura que acabem de veure, i a la dreta, separada per una distancia que es pot modificar, l'antena emissora (interrogador).

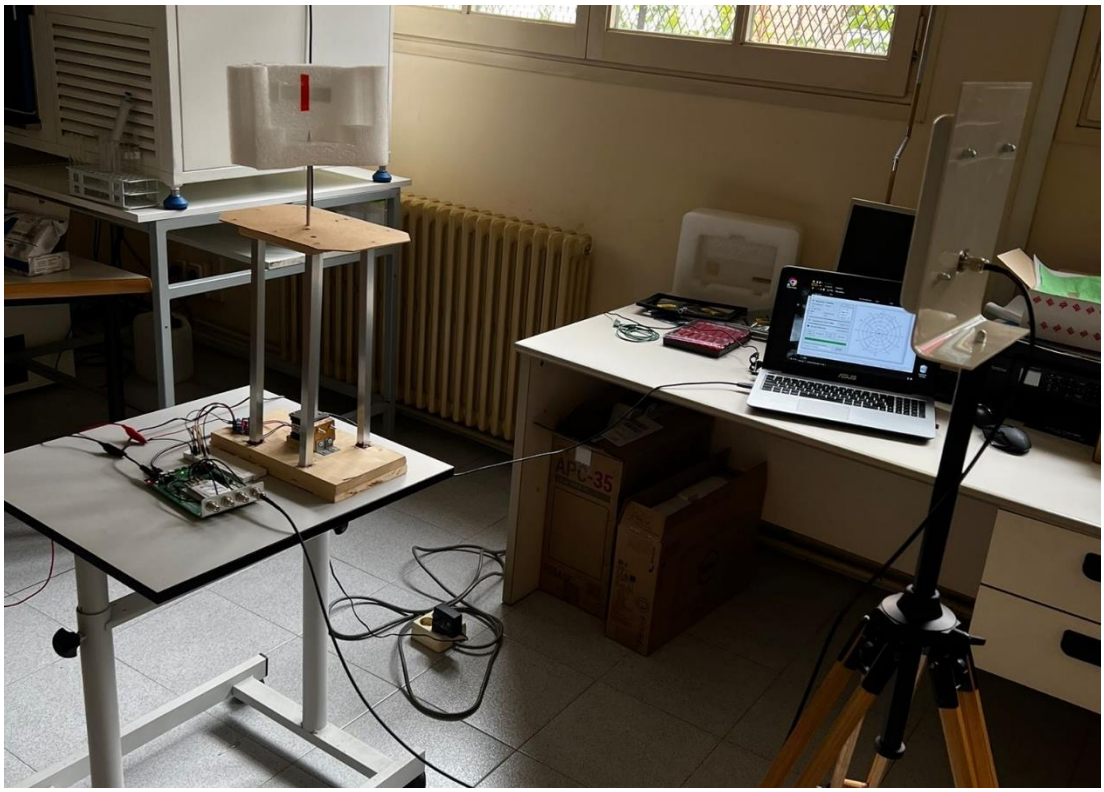

Figura 41. Sistema de posicionament real (Font: Fotografia pròpia)

4.2 Programa

Per al disseny i programació del software de control del sistema desenvolupat en aquest treball, així com per al de l'aplicació interfície, es va utilitzar el programa Visual Studio.

Com ja hem vist prèviament, aquest programa compta amb dos espais de treball, el del disseny gràfic de l'aplicació i el de la programació d'aquesta. És per això que en els següents apartats analitzarem per separat cada un d'aquests elements.

4.2.1 Disseny aplicació

A la figura 42 hi podem veure el disseny final de l'aplicació interfície amb la que els usuaris poden controlar el sistema de posicionament i mesura que hem vist al llarg d'aquest treball.

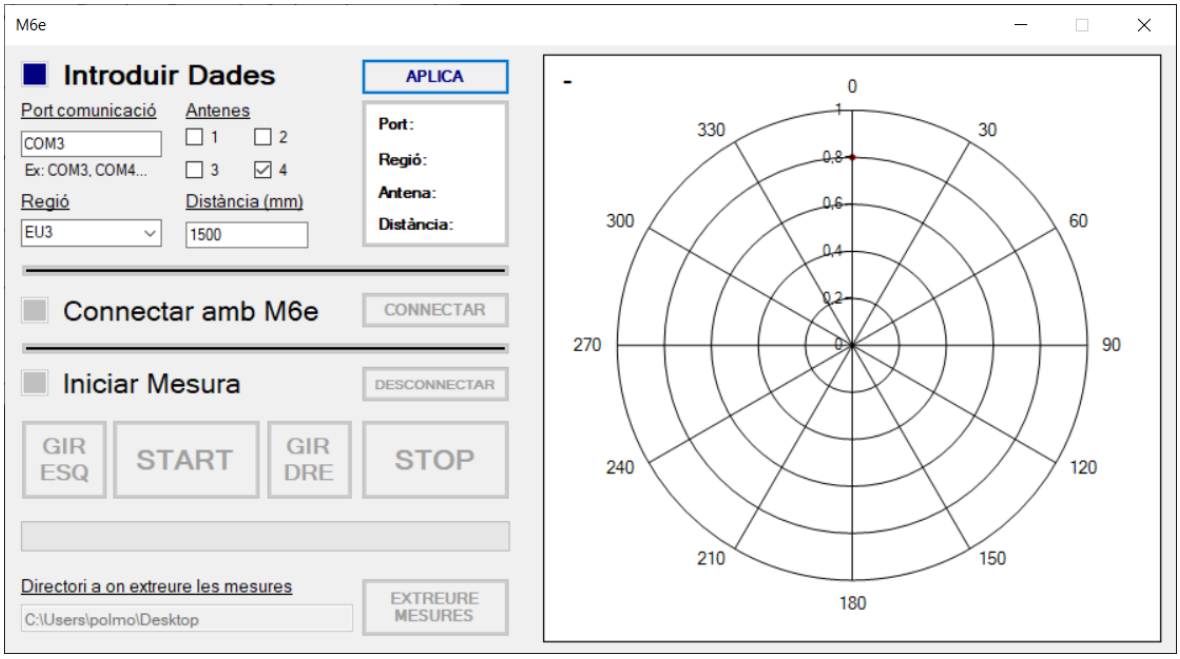

Figura 42. Aplicació interfície (Font: Captura pròpia)

D'entrada, són clarament identificables dues secciones diferenciades dins de la aplicació: a la part esquerra, formada per una varietat de botons i quadres de text, seria la que permet a l'usuari realitzar el control dels assajos, mentre que la de la dreta, formada exclusivament per un gràfic polar, serveix per a la visualització gràfica dels resultats.

Dins de la secció de "control" hi trobem 3 apartats diferents, identificats com a "Introduir dades", "Connectar amb M6e" i "Iniciar Mesura", els quals representen cada una de les fases en que es divideix un assaig.

L'apartat "Introduir Dades", permet als usuaris ajustar les condicions de l'assaig, podent seleccionar el port de comunicació que utilitza l'ordinador per a connectar amb el M6e (a traves d'un quadre de text on l'usuari hi ha d'escriure el port), la regió amb que es vol configurar el M6e per a l'assaig (a escollir entre una llista desplegable d'opcions com Europa, Nord Amèrica o Àsia), l'antena que s'està utilitzant en 'assaig (a traves dels quadres de selecció) i, finalment, la distancia en mil·límetres que hi ha entre els tag RFID a caracteritzar i l'antena emissora.

Un cop introduïdes aquestes dades, clicant el botó "APLICA", es verifica si les dades introduïdes són correctes, per exemple, com podem veure a la figura 43, si no es selecciona cap antena salta un missatge d'error indicant-nos que hem de seleccionar com a mínim 1.

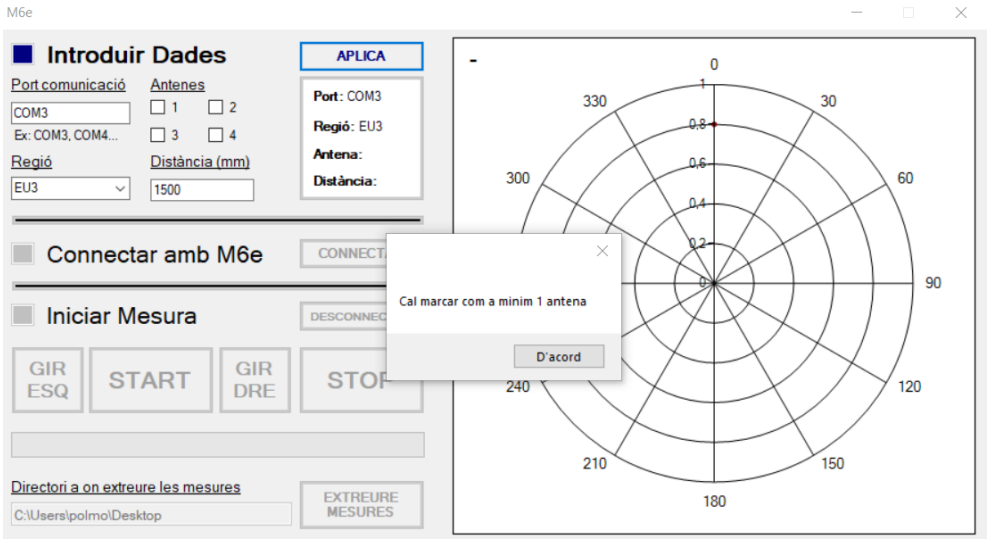

Figura 43. Error al introduir dades (Font: Captura pròpia)

Si les dades introduïdes con correctes, el programa habilita els elements de la següent secció "Connectar amb M6e" i deshabilita els de l'anterior. D'igual manera, com podem observar a la figura 44, requadre blau que prèviament indicava com a modificable el primer apartat, ha passat a indicar el segon, guiant així als usuaris d'una forma intuïtiva.

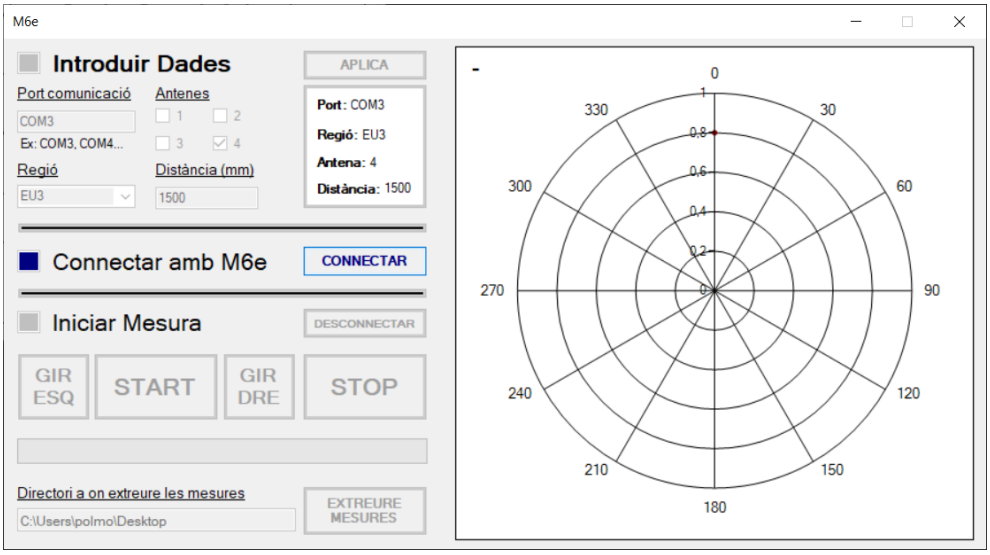

Figura 44. Apartat Connectar amb M6e (Font: Captura pròpia)

Dintre d'aquest apartat l'usuari només pot clicar el botó "CONNECTAR", amb el que s'intenta establir connexió amb el M6e. Si la connexió és efectiva, s'avançarà al següent apartat (figura 45), mentre que si sorgeix algun error, es generar un quadre de text especificant el que ha anat malament.

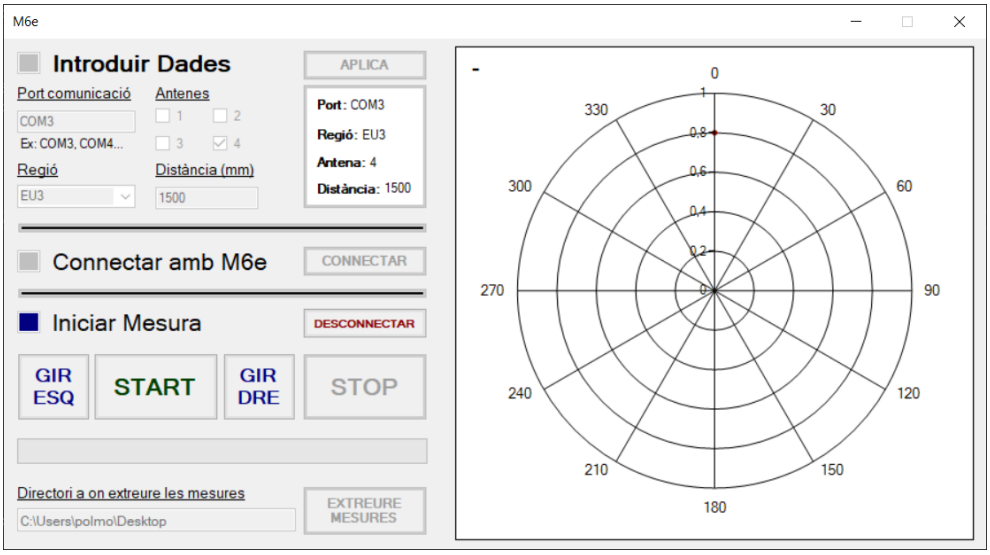

Figura 45. Apartat Iniciar Mesura (Font: Captura pròpia)

Novament, al avançar d'apartat, s'han activat els seus botons i s'ha "encès" l'indicador en forma de requadre blau. Dintre d'aquesta secció, l'usuari pot realitzar 4 accions diferents: ajustar la posició del motor fent-lo avançar o retrocedir amb els botons "GIR ESQ" i "GIR DRE", iniciar el procés de mesura amb el botó "START", o desconnectar-se del M6e clicant el botó "DESCONECTAR", amb el que es retorna a l'estat inicial, es a dir a la introducció de les dades.

Si es selecciona l'opció d'iniciar l'assaig durant a durada d'aquest només estarà habilitat el botó "STOP" (figura 46), el qual permet, en qualsevol moment de la mesura, realitzar una parada. Aquest element és especialment important ja que si sorgís algun problema tècnic durant la realització de les mesures seria de vital importància el ser capaç d'aturar el sistema, per, entre d'altres, evitar possibles danys als components electrònics. A part, durant el procés de mesures, també es pot observar en temps real com es va generant el gràfic, on s'hi afegeixen un a un els diferents punts mesurats, d'igual manera que es pot comprovar el progrés de l'assaig amb la barra de la part inferior.

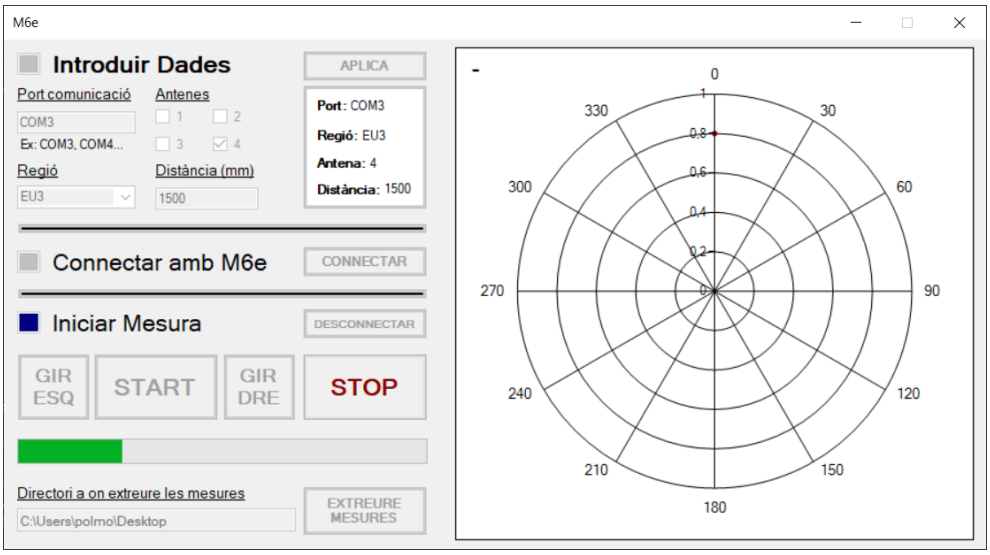

Figura 46. Assaig en progrés (Font: Captura pròpia)

Un cop finalitzat l'assaig, l'aplicació té un aspecte similar al de la figura 47, on a part del gràfic completat i la barra de progrés completa, quedem com a habilitats el ja esmentat botó "DESCONNECTAR" i una darrera secció que ens permet extreure les mesures realitzades en un fitxer txt. Per a realitzar aquesta acció nomes cal introduir el directori on es vol generar el fitxer en qüestió i clicar el botó "EXTREURE DADES".

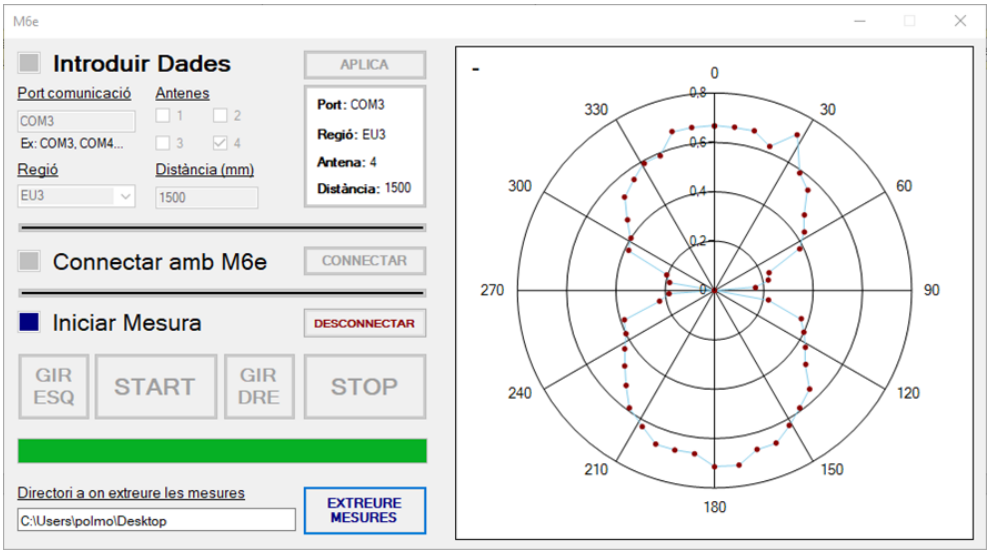

Figura 47. Fi de l'assaig (Font: Captura pròpia)

Com podem veure, una de les bases per a que l'aplicació sigui intuïtiva, alhora que evita possibles errors generats pels usuaris, és el fet d'activar i desactivar els botons que poden ser utilitzats per a cada estat, a part del fet de que dividir-la en diferents apartats permet als usuaris saber quina tasca estan realitzant en cada moment. No obstant, tot i que l'aplicació és força fàcil d'utilitzar, és va crear un breu manual d'instruccions per al seu us que podem veure tant de forma global a la figura 48 com en detall en l'Annex A d'aquest treball.

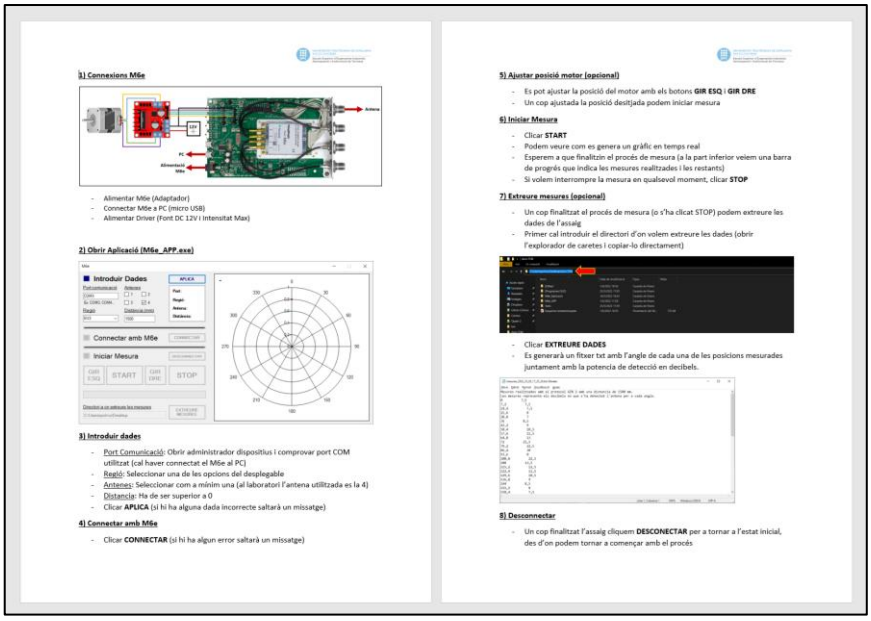

Figura 48. Manual instruccions (Font: Captura del manual realitzat en Word)

4.2.2 Programació

Al diagrama de la figura 49, hi podem observar el funcionament de l'aplicació tal i com ha quedat explicat a l'apartat anterior.

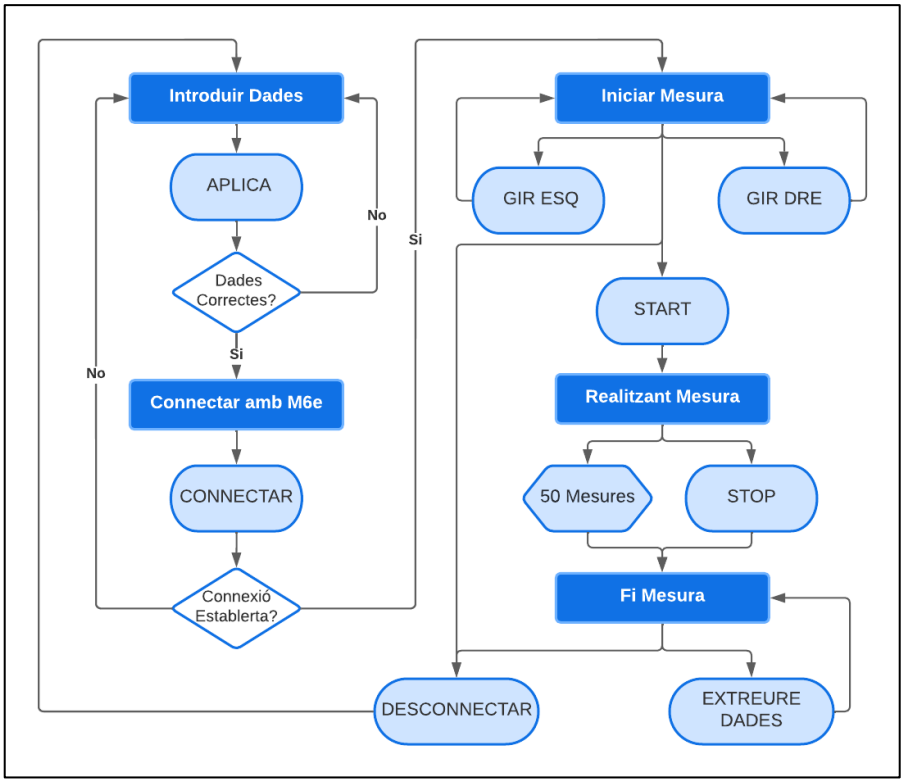

Figura 49. Diagrama flux aplicació (Font: Creat al web https://lucid.app/)

A l'hora d'aplicar un programa on en funció del estat es poden executar unes accions o altres, sempre es recomanable l'ús d'un element anomenat "Switch", el qual permet, a partir del valor d'una variable, executar un conjunt d'ordres o unes altres, actuant com una espècie de menú selector. Per aquest projecte, tenint en compte que, a part d'executar ordres en funció de quins botons es cliquen, també volem ajustar quins botons estan habilitats en cada instant, es va decidir utilitzar 2 "Switch" diferents, 1 per a realitzar les accions i un per a ajustar els elements de l'aplicació.

A part, el funcionament de Visual Studio, que com hem vist prèviament requereix de la introducció d'ordres per al clic de cada botó, va propiciar que cada un d'aquest menús selectors formessin els subprogrames "ACCIO_APP" i "ESTAT_APP", els quals s'invoquen cada cop que es vol realitzar alguna acció o ajustar l'estat de l'aplicació.

Finalment, i com veurem a continuació quan realitzem l'estudi en profunditat del codi, també va ser necessària la incorporació d'un subprograma addicional, anomenat "CODIFICA", el qual genera, a partir de la posició a la que volem ajustar el motor, una variable "gps" que permet a ajustar els ports GPIO requerits en cada moment.

4.2.2.1 Inicialització programa

Com podem veure a la figura 50, i com sol ser habitual, el codi desenvolupat en aquest projecte comença amb la declaració de les referencies de les llibreries a utilitzar. A part de les diferents llibreries estàndard del sistema, podem observat com també s'hi inclou la "ThingMagic", la qual conté el conjunt de les ordres necessàries per al control del M6e i que va ser facilitada en el paquet ja esmentat Mercury API.

```
//Declaracio 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;
```

Figura 50. Declaració de referencies (Font: Captura del codi desenvolupat)

Un cop fetes aquestes declaracions, iniciem el programa amb les ordres presents a la figura 51. A part d'inicialitzar els diferents elements de la aplicació, podem observar com s'ha afegit una ordre per a dibuixar el gràfic "net". Això es va fer per que, del contrari, aquest no apareixeria fins a l'obtenció de la primera mesura, deixant un requadre totalment blanc a la dreta de l'aplicació. D'aquesta manera, simplement dibuixem un pont "fals" el [0, 0.8] que posteriorment eliminarem abans d'introduir les mesures reals.

```
namespace ProbaAplica1
{
    3 references
    public partial class M6E : Form
    {

        1 reference
        public M6E()
        {
            InitializeComponent();
            chart.Series["F1"].Points.AddXY(0, 0.8); //Iniciem el programa dibuixant el grafic "net"
        }
```

Figura 51. Inicialització programa (Font: Captura del codi desenvolupat)

Després d'inicialitzar el programa, es realitza un altre procediment habitual, la declaració de les variables globals que s'utilitzaran al llarg del programa. Com observem a la figura 52, la majoria de les variables son "int" (nombres enters), "string" (variables alfanumèriques" i "bool" (booleans que nomes poden tenir els valors 1 o 0). A part també s'hi inclouen un "double" (nombre decimal), una llista d'aquest mateixos elements i un element "TagReadData" element que necessita el M6e per a emmagatzemar les lectures de tags.

```
//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;
```

Figura 52. Declaració de variables (Font: Captura del codi desenvolupat)

4.2.2.2 Subprograma "ACCIO_APP"

Amb això fet, el programa procedeix amb el subprograma ja explicat "ACCIO_APP", el qual executa un conjunt d'ordres en funció de la variable entera "ACCIO" utilitzant un "Switch". A la figura 53 podem veure aquest subprograma amb els diferents casos (accions a realitzar) comprimits, els quals detallarem a continuació.

```
// Subprograma ACCIO_APP: Executa les diferents ordres en funció de la variable "ACCIO"
7 references
private async void ACCIO_APP()
{
    switch (ACCIO)
    {
        case 0: //S'ha clicat aplica dades
            ...
            break;

        case 1: //S'ha clicat connectar amb m6e
            ...
            break;

        case 2: //Gir motor Esquerra
            ...
            break;

        case 3: //Gir motor Dreta
            ...
            break;

        case 4: //S'ha clicat inicia mesura
            ...
            break;

        case 5: // S'ha clicat extreure dades
            ...
            break;

        case 6: // S'ha clicat desconectar de m6e
            ...
            break;

        default:
            break;
    }
}
```

Figura 53. Subprograma "ACCIO_APP" (Font: Captura del codi desenvolupat)

4.2.2.2.1 Cas 0 - Aplicar dades

Quan s'executa el subprograma amb la variable "ACCIO" valent 0 (situació a la que arribem després de clicar el botó "Aplicar dades", aquest primerament "neteja" les variables que utilitzarà, posant-les a 0 (figura 54).

```
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;
```

Figura 54. Inici cas 0 de "ACCIO_APP" (Font: Captura del codi desenvolupat)

Seguidament realitza les comprovacions de que les dades introduïdes en els requadres de port de comunicació i regió no siguin nul·les. Amb les ordres de la figura 55, fem principalment 2 accions: si les dades son correctes augmentem dues vegades la variable dades (indicant que ja en tenim dues), mentre que si són incorrectes generem un quadre de diàleg indicant l'error.

```
//Comprobem que s'hagi introduit 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");

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

Figura 55. Comprovació dades Port i Regió (Font: Captura del codi desenvolupat)

A continuació es verifiquen i processen les dades relatives a les antenes. En tant que aquestes dades s'introdueixen amb quadres de selecció cal utilitzar el conjunt d'ordres de la figura 56, per a generar el vector d'antenes a utilitzar (si es que n'hi ha alguna de seleccionada). Novament, si les dades són les adients es s'augmenta la variable dades, i si no ho són, es crea el quadre de diàleg.

```
//Comprobem que s'hagi introduit com a minim 1 antena
if (cbx_ant1.Checked)
{
    if ((cbx_ant2.Checked) || (cbx_ant3.Checked) || (cbx_ant4.Checked)) 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 necesita el M6e amb les antenes seleccionades
    lbl_antena.Text = llista;
    dades++;
}
else MessageBox.Show("Cal marcar com a minim 1 antena");
```

Figura 56. Comprovació dades Antena (Font: Captura del codi desenvolupat)

La darrera dada a verificar és la distancia. En tal que aquesta ha de ser un enter en mil·límetres superior a 0, es realitzen dues comprovacions: primerament s'intenta de convertir el text introduït en un enter (utilitzant les ordres "try – catch") de tal manera que, si per exemple s'ha escrit una lletra, es generi un nou quadre de text indicant l'error.

Seguidament, també es verifica que la distancia introdudida sigui superior a 0. Una altra vegada, si el procés de validació és efectiu, s'augmenta la variable dades.

```
//Comprobem que la distancia introduida 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 distancia");
    else MessageBox.Show("Distancia introduida en format incorrecte, enter superior a 0." + ex1.Message);
}
```

Figura 57. Comprovació dades distancia (Font: Captura del codi desenvolupat)

Finalment, amb les ordres de la figura 58, es comprova el valor de la variable dades. Si aquest és igual a 4, significa que totes han esta introduïdes en un format adient, pel que es pot avançar a l'estat 1 (Connectar amb M6e). Pel contrari, si la variable té un valor inferior, significa que alguna de les dades introduïdes és incorrecta, pel que es manté l'estat de l'aplicació (Introduir dades).

```
    //Comprobem si s'han introduit totes les dades necesaries
    if (dades == 4)
    {
        ESTAT = 1;
        ESTAT_APP();
    }
}
break;
```

Figura 58. Final cas 0 de "ACCIO_APP" (Font: Captura del codi desenvolupat)

4.2.2.2.2 Cas 1- Connectar amb M6e

Quan s'executa el subprograma amb la variables "ACCIO" amb un valor de 1, s'estableix de manera temporal l'estat de la aplicació al 5 (el qual, com veurem mes endavant, desactiva tots els botons), per tal d'evitar que l'usuari pugui afectar l'intent de connectar amb el M6e, i seguidament es desactiven el conjunt de booleans a utilitzar.

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

        STOP = false;
        bucle = false;
        conectat = false;
```

Figura 59. Inici cas 1 de "ACCIO_APP" (Font: Captura del codi desenvolupat)

A continuació, com podem veure a la figura 60, s'utilitza un "try" per a intentar connectar amb el M6e amb les ordres que hem vist prèviament: la de crear l'objecte reader i la "r.Conect". Seguidament, definim els ports GPIO com a sortides digitals i els ajustem a la posició inicial (1). Com es pot observar primerament s'ajusten a las posició 1 amb la variables POS, però després de de 250 milisegons es desactiven posant-se a la posició 0. Això es deu al sobreescalfament del driver explicat prèviament, el qual va trobar com a solució el fet de només alimentar el motor a l'hora de posicionar-lo i no enclavant-lo en els temps intermedis.

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

        //Ajustem ports GPIO a la posicio 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));
```

Figura 60. Connexió amb M6e i ajust GPIO (Font: Captura del codi desenvolupat)

Seguidament, amb les ordres de la figura 61 es tanca l'intent de connexió comprovant si les antenes són detectades i ajustant la regió seleccionada. Com es feia prèviament, si es produís algun error durant el procés es generaria un quadre de diàleg indicant-ho, mentre que si la connexió és efectiva el booleà "connectat" prendrà valor 1.

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

Figura 61. Comprovació antena i ajust regió (Font: Captura del codi desenvolupat)

Finalment, amb les ordres de la figura 62, es finalitza el procés de connexió comprovant el valor del booleà "connectat". Si aquest és cert s'ajusta l'estat al 2 (Iniciar Mesura), i si és fals es retorna a l'inicial 0 (Introduir dades).

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

Figura 62. Final cas 1 de "ACCIO_APP" (Font: Captura del codi desenvolupat)

4.2.2.2.3 Cas 2- Gir motor esquerra

Quan es clica el botó "GIR ESQ", la variable ACCIO pren el valor 2, pel que s'executen les ordres presents a la figura 63. Com podem veure, primerament s'ajusta l'estat de l'aplicació al cinquè, que com hem vist prèviament desactiva tots els elements d'aquesta, per a seguidament connectar-se amb el M6e. Un cop establerta la connexió, es mou el motor en 4 posicions disminuint el valor de la variable "POS" (cal recordar que el motor pot prendre 200 posicions, i al voler realitzar 50 mesures, cal separar-les per 200/50=4 posicions), repetint el procediment vist prèviament de desactivar el ports GPIO (ajustat la posició 0) un cop ajustat el motor per a evitar el sobre escalfament del driver. Un cop realitzat el procediment, es torna a ajustat l'estat de la App al 2 (Iniciar Mesura).

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

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

            //Disminuim la posicio 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;
```

Figura 63. Cas 2 de "ACCIO_APP" (Font: Captura del codi desenvolupat)

4.2.2.2.4 Cas 3- Gir motor dreta

Com podem observar a la figura 64, el cas 3 d'aquest subprograma és idèntic al anterior, amb l'única diferencia que, per a girar el motor cap a la dreta, en lloc de disminuir la variable "POS" en cada intercanvi de posició, n'augmentem el valor.

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

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

            //Aumentem 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;
```

Figura 64. Cas 3 de "ACCIO_APP" (Font: Captura del codi desenvolupat)

4.2.2.2.5 Cas 4- Inici mesura

Per a poder comprendre les ordres que executa el programa quan el valor de la variable ACCIO és 4, situació a la que s'arriba després de clicar el botó "START", primer cal comprendre en que consisteix aquest el procés de mesura. Aquest el podem veure explicat al diagrama de flux de la figura 65.

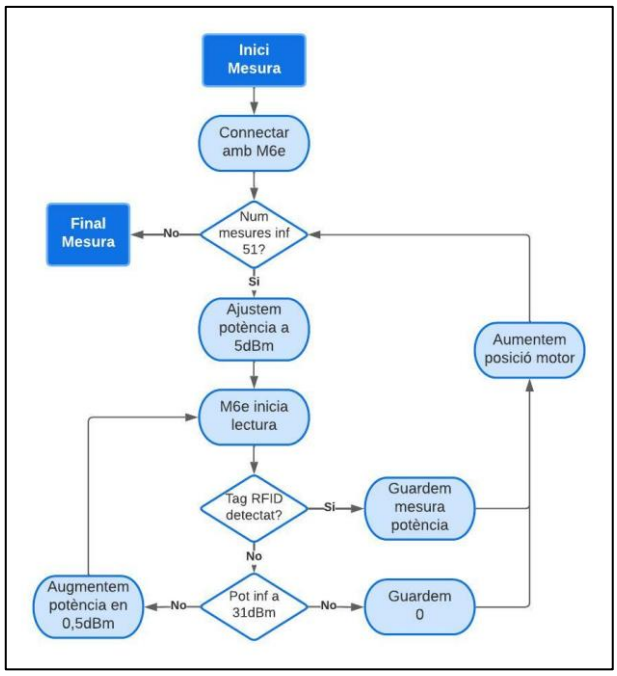

Figura 65. Diagrama flux procés mesura (Font: Creat al web)

Primerament es connecta amb el M6e, i mentre el numero de mesures sigui inferior a 51 (en lloc de realitzar només les 50 mesures indicades, realitzem dos cops la de l'angle 0, per això poder verificar que no hi ha hagut desviacions al llarg de l'assaig) es repeteix un procediment consistent en ajustar la potencia de lectura al mínim (5 dBm) i anar-la augmentant (amb intervals de 0,5 dBm) fins a que el tag és detectat, moment en que es desa la mesura de la potencia de detecció. Si el tag no arribes a ser detectat un cop arribat a la potencia màxima (31 dBm), es desaria com a mesura un 0, indicant la no detecció. En qualsevol d'aquests dos casos és procediria a augmentar la posició del motor (com hem vist abans 4 vegades), per a tornar a verificar que el nombre de mesures no ha arribat a 51, cas en que l'assaig finalitzaria.

Per a realitzar això, primerament ajustem l'estat de la aplicació al 3, on l'únic botó operatiu és el de parada d'emergència "STOP", juntament amb l'ajustament i neteja de les variables a utilitzar. Tal i com s'observa a la figura 66, al inici d'aquest cas també realitzem la ja habitual connexió amb el M6e (creant l'objecte reader i amb l'ordre "r.Connect") juntament amb la definició del protocol de lectura, el qual és l'estàndard "GEN2".

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

        chart.Series["F1"].Points.Clear();//Netejem grafic abans de dibuixar
        BarraP.Value = 0;//Netejem barra progres
        STOP = false; //Desactivem bool stop
        bucle = true; //Activem bool bucle
        llista_mesures.Clear(); //Netejem 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, TagProtocol.GEN2, null, null, 1000)); //Ajustem el protocol de mesura
```

Figura 66. Inici cas 4 de "ACCIO_APP" (Font: Captura del codi desenvolupat)

Seguidament, amb les ordres de la figura 67, el programa inicia el procés de les 51 mesures, desactivant el booleà "detectat" (el qual només serà activat un cop trobada la potencia de detecció dels tags) i ajustant la potencia inicial a 5 dBm. D'igual manera, amb el "While" (element que permet repetir un conjunt d'ordres mentre es compleixi una expressió lògica) procedim a augmentar la potencia de lectura mentre aquesta sigui inferior a la màxima (31 dBm) i no s'hagi detectat cap tag, comprovació que és realitza verificant si la longitud de la llista de tags detectats és superior a 0.

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

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

Figura 67. Mesura de tags (Font: Captura del codi desenvolupat)

Amb això fet, el programa procedeix verificant el valor de la variable booleana "detectat" (figura 68), de tal manera que si aquesta és certa es desa a la variable "calcul_formula" i és grafica la potencia de detecció dividida per la mínima de 5 dBm (d'aquesta manera el gràfic resultant s'hi mostren dades en tant per 1, on 1 equival a una potencia de detecció mínima, i valors pròxims a 0, potencies més elevades). Si pel contrari la variable és falsa, es grafica i es desa a "calcul_formula" un 0. Finalment, i cop hem fet prèviament a les ordres de "Gir Motor", tornem a augmentar en 4 unitats la posició del motor per a posteriorment desactivar els ports GPIO.

```
//Si despres de realitzar la mesura s'ha detectat el TAG, es grafica el punt i s'afegeix la potencia de deteccio 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 divisio de 5 dB per la potencia de deteccio
}
//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);
}

//Despres de cada mesura augmenem la posicio del motor en 4 unitats (200 posicions del motor entre 4 equival a realitzar 50 mesures)
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));
```

Figura 68. Post mesura de tags (Font: Captura del codi desenvolupat)

Finalment, i com podem observar a la figura 69, el programa afegeix a la llista de mesures els valors de la variable "Calcul_formula" que acabem de veure i augmenta el valor de la barra de progrés de la aplicació. D'igual manera, un cop realitzades les 51 mesures, el programa s'assegura de que els ports GPIO han quedat desactivats tornant a ajustar-los a la posició 0, per a finalitzar ajustant l'estat de la aplicació al 4, on l'usuari pot extreure dades o desconnectar del M6e.

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

Figura 69. Final cas 4 de "ACCIO_APP" (Font: Captura del codi desenvolupat)

4.2.2.2.6 Cas 5- Extreure dades

El cinquè cas del subprograma, amb el que els usuaris poden extreure un fitxer ".txt" amb les dades mesurades, utilitza les ordres de la figura 70 per a, primerament, generar el fitxer on es desaran les dades al directori introduït per l'usuari, procés que requereix verificar que el directori realment existeixi. Un cop creat aquest fitxer, el qual té al nom la data i l'hora de la mesura per poder ser fàcilment identificable, el programa hi afegeix primer un parell de línies explicatives de les condicions de l'assaig (distancia i protocol utilitzat) així com una breu descripció del format de les dades. Fet això es procedeix a introduir cada una de les dades de la llista creada al assaig, indicant per a cada mesura el seu angle corresponent.

```
case 5: // S'ha clicat extreure dades
    {
        //Primer comprovem que el directori on extreurem les dades existeix
        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";
            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 represente els decibels en que s'ha detectat l'antena per a cada angle.");
            //Afegim a l'arxiu cadauna 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 existeig");
    }
    break;
```

Figura 70. Cas 5 de "ACCIO_APP" (Font: Captura del codi desenvolupat)

4.2.2.2.7 Cas 6- Desconnectar

El darrer cas del subprograma, el 6, és executat quan l'usuari clica el botó "DESCONNECTAR", i té la funció de tornar l'aplicació al estat inicial per a la realització d'un nou assaig. Com veiem a la figura 71, primer s'ajusta temporalment l'estat de la aplicació al 5 (on tots els botons estan deshabitats), per a procedir amb la "neteja" del gràfic creat i de la barra de progres, la qual torna a marcar un 0%. Seguidament és torna a verificar que els ports GPIO quedin desactivats ajustant-ne la posició al 0, per a finalitzar definint com a estat de la App l'inicial (Introduir Dades).

```
        case 6: // S'ha clicat desconectar de m6e
            {

                ESTAT = 5;
                ESTAT_APP();

                chart.Series["F1"].Points.Clear(); //Netejem grafic
                chart.Series["F1"].Points.AddXY(0, 0.8);//Dibuixem grafic "net"
                BarraP.Value = 0;//Netejem 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;
    }
}
```

Figura 71. Cas 6 de "ACCIO_APP" (Font: Captura del codi desenvolupat)

4.2.2.3 Subprograma "ESTAT_APP"

Un cop finalitzat el subprograma "ACCIO_APP", el codi procedeix amb un altre dels subprograma explicats prèviament, el "ESTAT_APP". Com podem veure a la figura 72, aquest també esta format per un "Switch" el qual s'utilitza de la variable "ESTAT" per a definir quins elements de la aplicació han d'estar o no habilitats.

```
// Subprograma ESTAT_APP: Habilita o deshabilita els elements de la aplicacio (botons clicables, textos modificables...) en fincio de la variable "ESTAT"
14 references
private void ESTAT_APP()
{
    switch (ESTAT)
    {
        case 0: // S'han d'introduir dades i clicar aplica dades
            ...
            break;

        case 1: // S'han verificat les dades i cal clicar connectar
            ...
            break;

        case 2: // S'ha conectat amb el M6e, cal iniciar mesura, ajustar angle motor o desconectar
            ...
            break;

        case 3: // S'esta realitzant mesura
            ...
            break;

        case 4: // Ha finalitzat la mesura o s'ha clicat boto STOP, cal extreure dades o desconectar
            ...
            break;

        case 5: // No es pot clicar res, el programa esta treballant
            ...
            break;

        default:
            break;
    }
}
```

Figura 72. Subprograma "ESTAT_APP" (Font: Captura del codi desenvolupat)

Com es pot observar a la figura anterior, la variable "ESTAT" pot prendre els 6 valors diferents:

- Cas 0 – Estat inicial "Introduir Dades"
- Cas 1 – Estat "Connectar amb M6e"
- Cas 2 – Estat "Iniciar Mesura"
- Cas 3 – Estat "Mesura en progres" (on només esta habilitat el botó "STOP")
- Cas 4 – Estat "Mesura Realitzada"
- Cas 5 – Estat "Desactivat" (on tots els elements estan deshabitats)

En tant que tots els 7 estats tenen una estructura idèntica, simplement modificant els valors assignats a cada element (habilitant-los amb un "true" o deshabilitant-los amb un "false"), només analitzarem en detall un d'ells, per exemple el cinquè, on tots els elements han d'estar deshabitats.

Com podem veure a la figura 73, primer s'ajusten els colors dels rectangles indicadors, en aquest cas, com només es pot accedir a aquest estat des de un procés de mesura, l'únic il·luminat és el "Ind_mesura" el qual es pinta de color blau (en concret el to "Navy"), mentre que la resta es deixen en gris. A continuació es deshabilitant un per un tots els elements de la aplicació, assignant com a falsos els seus "enabled": primerament els quadres de text, per a seguir amb el selector de regió, les caselles de selecció de les antenes i finalment els diferents botons.

```
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;
```

Figura 73. Cas 5 de “ESTAT_APP” (Font: Captura del codi desenvolupat)

4.2.2.4 *Definició botons*

Un cop finalitzat aquest darrer subprograma, el codi continua amb les definicions de les accions a realitzar després del clic de cada botó. Com s'observa a les figura 74, per a cada un dels botons s'utilitza el mateix conjunt d'ordres: primer s'ajusta el valor de la variable "ACCIO" per a seguir amb la "crida" del subprograma "ACCIO_APP". És aquí on es pot observar la gran efectivitat de l'ús d'un únic subprograma amb un "Switch", ja que del contrari, s'haurien d'haver afegit les diferents ordres vistes prèviament dins de les diferents crides al botons, pel que el procés de programació hauria estat força més complicat.

```
// Accions a executar quan es clica un boto

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

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

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

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

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

1 reference
private void btn_desconect_Click(object sender, EventArgs e)
{
    ACCIO = 6;
    ACCIO_APP();
}// boto desconecta
```

Figura 74. Definicions botons (Font: Captura del codi desenvolupat)

A part dels diferents botons "normals" de la figura anterior, el programa també compta amb un botó que té un funcionament especial. Es tracta del "STOP", el qual te la capacitat d'aturar el procés de mesura en qualsevol moment. Com podem veure a la figura 75, aquest primerament ajusta l'estat de la aplicació al 5 (que com hem vist prèviament deshabilita tots els elements) per a procedir amb la activació d'un booleà que atura el procés de mesura. Per a aconseguir aquest efecte, el subprograma "ACCIO_APP" es va definir com a "asíncron" fet que permet el clic de botons durant la seva execució. A part, el booleà esmentat, apareix dins del "bucle" de 51 mesures, sent una de les condicions per a que aquest pugui continuar, de tal manera que quan aquest pren un valor positiu, el "bucle" finalitza tot i no haver realitzat el total de mesures.

```
//El boto STOP es especial, pot interrompre el bucle de mesures
1 reference
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();
}
```

Figura 75. Definicions botó "STOP" (Font: Captura del codi desenvolupat)

A la figura anterior també es pot observar com el clic del botó "STOP" s'assegura de que els ports GPIO quedin desactivats, després del qual ajusta l'estat de la aplicació al 4 (Mesura Realitzada).

4.2.2.5 Subprograma "CODIFICA"

El darrer element que apareix en el codi d'aquest software de control, és el ja explicat subprograma "CODIFICA", que te com a funció la de generar la variable "gps" per a ajustar els ports GPIO en funció del valor de la variable "POS".

Com ja hem vist prèviament, per al control del motor del sistema desenvolupat, es va decidir utilitzar la seqüència del "Double Step", per a augmentar el parell motor i així assegurar el moviment del sistema mecànic.

A la figura 76 hi podem observar l'inici d'aquest subprograma, on simplement és crea la variable "gps" i es defineixen les ordres a executar en el cas de que la posició desitjada sigui la 0 (tots els ports desactivats). Com veiem, primer es creen els 4 elements "GpioPin" corresponents a cada port, als quals se'ls assigna un numero de port i un estat (on false equival a desactivat i true a activat). Amb aquest elements generats, només cal afegir-los al "gps" que serà utilitzat per a ajustar els ports i així controlar el motor.

```
//Subprograma CODIFICA: entrem amb posicio (1,2,3,4 o 0) i retorna variable "gps" necesaria per a executar set GPIO
10 references
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);
    }
```

Figura 76. Inici subprograma "CODIFICA" (Font: Captura del codi desenvolupat)

El programa procedeix amb les definicions de les 4 posicions restants, seguint un format idèntic al de la 0, amb l'única diferencia dels ports a activar en cada una. A la figura 77 és poden observar aquestes definicions, les quals segueixen la seqüència explicada en apartats anteriors (cal tenir present que el port 1 esta vinculat al born A+ del motor, el 2 al A-, el 3 al B+ i el 4 al B-).

```
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);
}
```

Figura 77. Posicions 1 a 4 subprograma "CODIFICA" (Font: Captura del codi desenvolupat)

Finalment, el codi del software finalitza amb la fi del subprograma, que retorna el valor de la variable "gps" generada (figura 78).

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

Figura 78. Fi del subprograma "CODIFICA" i del codi (Font: Captura del codi desenvolupat)

5 Resultats

Un cop finalitzat el disseny del sistema, es va procedir amb la validació d'aquest, procés per al que es van realitzar les caracteritzacions de 3 tags RFID diferents, entre els que hi destaca un dels desenvolupats al departament elèctric de la ESEIAAT.

Abans de procedir amb els assajos, cal destacar una característica dels tags RFID, i es que, degut al disseny de les antenes d'aquests, la potencia de detecció no només depèn l'angle relatiu entre l'antena emissora i el tag, sinó també de la posició d'aquest últim. El diagrama de radiació d'un tag en posició "horitzontal" és el que té la forma de 8 que havíem vist prèviament, mentre que si aquest és col·locat de manera vertical, la forma que genera és la d'un cercle. Degut això, per a cada un dels tags avaluats, es van realitzar dues mesures, una per a cada una d'aquestes posicions.

El primer tag analitzat (figura 79) va ser un dels inclosos en el kit de desenvolupament M6e, el qual és el típic tag en forma d'etiqueta adhesiva que podem trobar a diferents productes d'un supermercat.

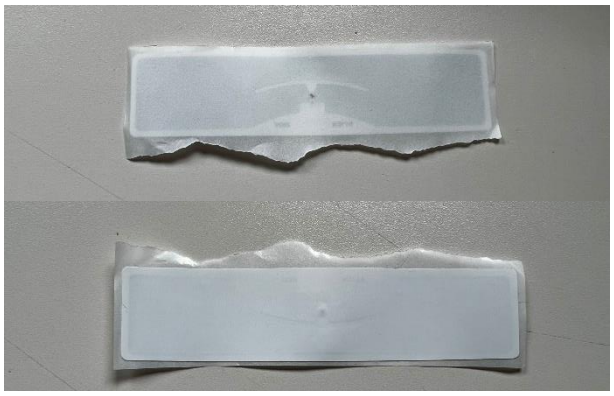

Figura 79. Tag RFID 1 (Font: Fotografia pròpia)

Com hem esmentat, per a cada tag es van realitzar mesures en dues posicions, horitzontal i vertical, la primera de les quals podem veure a la figura 80.

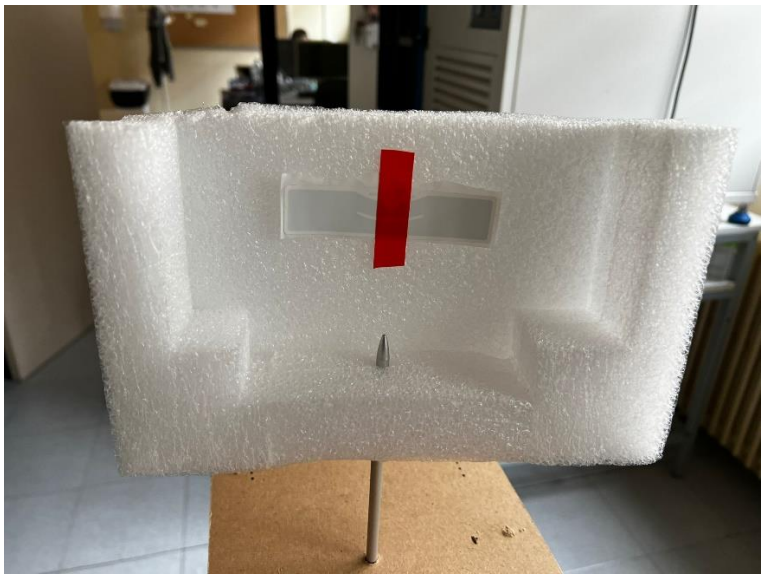

Figura 80. Tag RFID 1 posició horitzontal (Font: Fotografia pròpia)

Un cop realitzada la caracterització d'aquest primer tag en la posició horitzontal és va obtenir el diagrama de radiació de la figura 81. Com podem observar, el resultat obtingut s'ajusta a la perfecció al desitjat, formant un 8 quasi perfecte. No obstant cal destacar que per a les 2 posicions perpendiculars (90 i 270), es pot observar com les mesures graficades són zeros, indicant que en aquells punts la potencia de detecció va ser superior a 31 dBm i per tant no va ser detectat pel sistema.

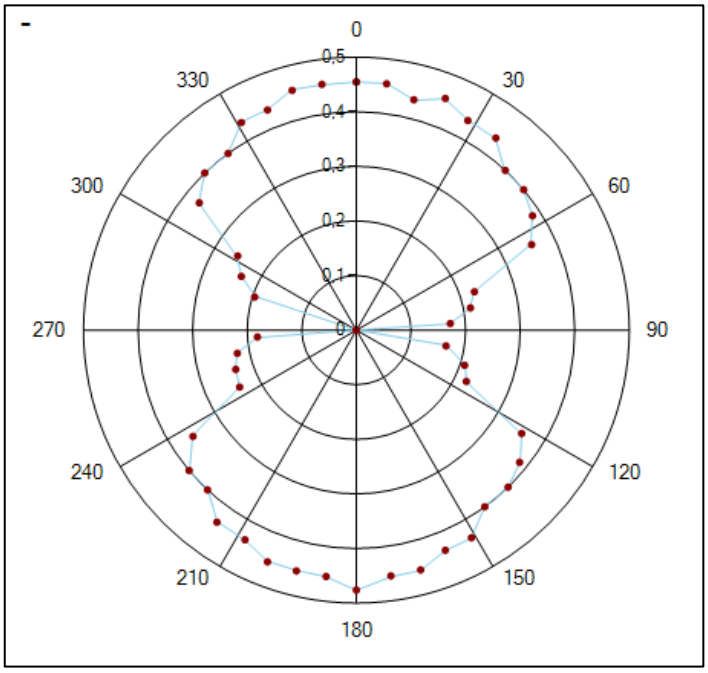

Figura 81. Diagrama radiació tag RFID 1 posició horitzontal (Font: Captura pròpia)

Per aquest mateix tag, es va realitzar seguidament la caracterització en la posició vertical (figura 82).

Figura 82. Tag RFID 1 posició vertical (Font: Fotografia pròpia)

Els resultats d'aquest segon assaig es poden veure a la figura 83, amb la que podem verificar la forma circular del diagrama de radiació del tag en la posició vertical. D'aquesta mesura, no obstant, cal destacar la petita desviació present en la 4 posició, la qual, al no repetir-se posteriorment, fa pensar que no és més que un error puntual en la mesura.

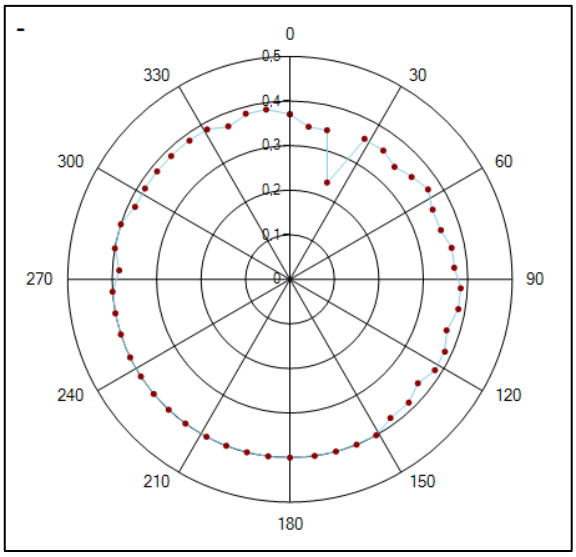

Figura 83. Diagrama radiació tag RFID 1 posició vertical (Font: Captura pròpia)

El següent tag avaluat, va ser un altre dels inclosos en el kit de desenvolupament M6e, el qual podem veure a la figura 84 amb la forma de tarja d'accés a una universitat.

Figura 84. Tag RFID 2 (Font: Fotografia pròpia)

Per a aquest tag es van realitzar novament les mesures en dues posicions, les quals podem veure a la figura 85.

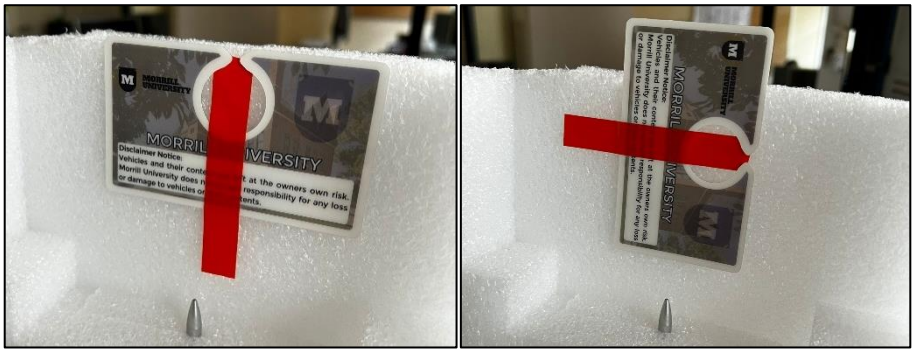

Figura 85. Tag RFID 2 posicions horitzontal i vertical (Font: Fotografia pròpia)

Els resultats d'aquest tag, es poden observar en la figura 86, on veiem els diagrames de radiació tant en la posició horitzontal com en la vertical. Del primer no hi ha grans elements a destacar, té una forma pròxima al 8 i al igual que amb l'anterior, les posicions perpendiculars generen no deteccions (zeros), en canvi, és analitzant el segon (posició vertical) que detectem un comportament estrany. Si bé la forma és pròxima al cercle, es pot observar com aquesta circumferència no està del tot centrada. Una possible explicació és que l'interior del tag no sigui simètric, i la bobina que forma la seva antena modifiqui lleugerament la seva àrea de radiació al llarg del gir.

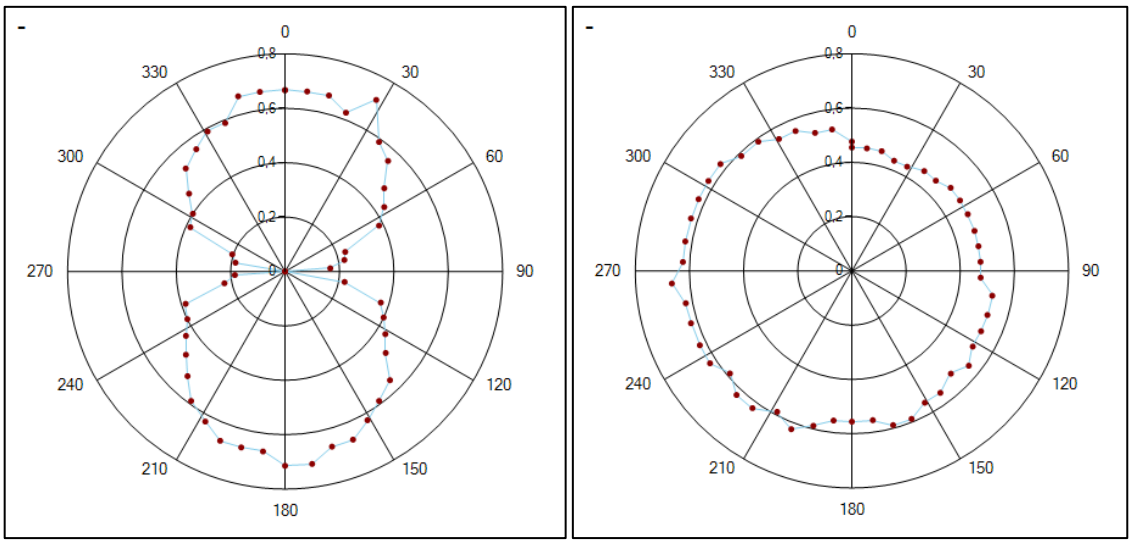

Figura 86. Diagrames radiació tag RFID 2 posicions horitzontal i vertical (Font: Captura pròpia)

El darrer tag avaluat en aquest procés de verificació, com hem comentat, va ser un dels desenvolupats a la ESEIAAT amb teixit (figura 87).

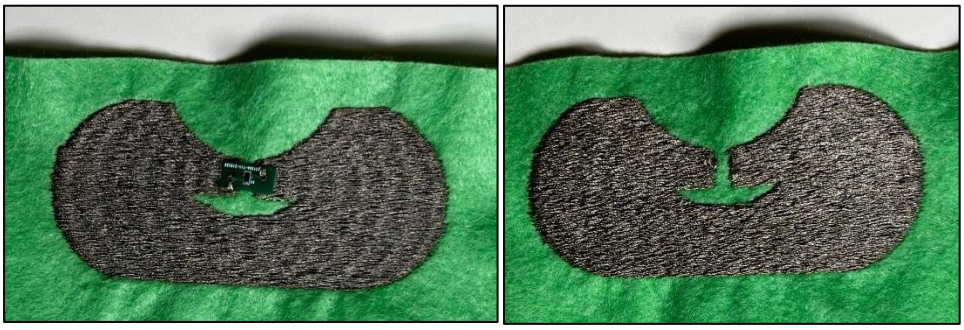

Figura 87. Tag RFID 3 (Font: Fotografia pròpia)

Igual que per als anteriors, es van realitzar les caracteritzacions per les dues posicions, tal i com podem veure a la figura 88.

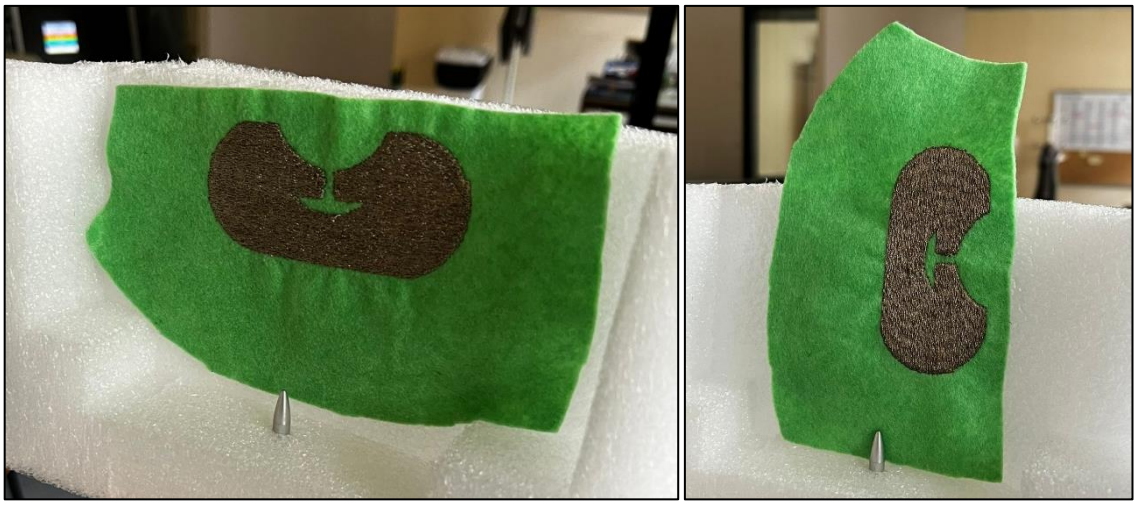

Figura 88. Tag RFID 3 posicions horitzontal i vertical (Font: Fotografia pròpia)

D'aquesta manera, els resultats obtinguts en les deus posicions poden ser observats a la figura 89.

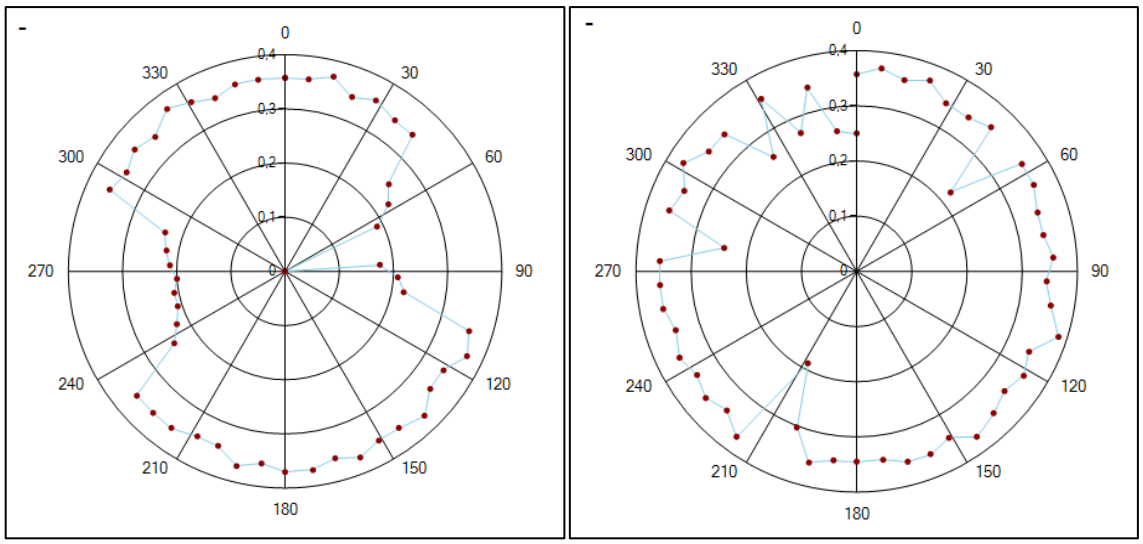

Figura 89. Diagrames radiació tag RFID 3 posicions horitzontal i vertical (Font: Captura pròpia)

Com podem observar, la forma del diagrama de la posició horitzontal no és tant un 8, sinó que més aviat s'aproxima a la d'una destral. Això, no és un indicador d'un funcionament erroni de l'aparell, sinó simplement una indicació de que en efecte el tag desenvolupat a l'ESEIAAT té un diagrama de radiació diferent.

On si que es poden observar alguns patrons anormals és al analitzar la posició vertical ,la qual, si bé en general s'aproxima al cercle desitjat, presenta unes desviacions especialment marcades en els angles 50 i 210. En tant que la desviació es repeteix d'una forma gairebé simètrica, tot sembla indicar que aquesta és deguda al propi tag, tot i que d'igual manera es podria tractar d'un error puntual.

6 Resum del pressupost

Els costs a avaluar per aquest projecte, es poden classificar en costs de materials adquirits per al desenvolupament del sistema de posicionament, i en els generats per les hores de treball invertides en les diferents tasques realitzades en aquest projecte.

Com es pot veure en detall al document annexat "Pressupost", els totals d'aquests costs són els següents:

Concepte	Cost
Costs materials	1.395,77€
Costs hores de treball	5.475,00 €
Total	**6.870,77 €**

Figura 90. Total costs del projecte (Font: Taula pròpia)

7 Conclusions

Un cop finalitzat el procés d'assajos de validació del sistema desenvolupat podem concloure amb el següent:

- S'ha assolit el principal objectiu d'aquest projecte de desenvolupar el sistema de posicionament i caracterització d'antenes RFID.
- S'ha aconseguit controlar el kit de desenvolupament M6e per a de poder obtenir la potencia de detecció d'antenes RFID i amb els seus ports GPIO controlar un motor pas a pas per a ajustar la posició d'antenes RFID.
- S'ha aconseguit que la aplicació desenvolupada permeti als usuaris ajustar les dades de l'assaig (regió, port de comunicacions, antena utilitzada i distancia entre el sistema de mesura i l'antena a mesurar), això com ajustar la posició inicial del motor controlat.
- S'ha aconseguit que el sistema desenvolupat realitzi 51 mesures (repetint dues vegades la inicial) separades per 7,2º (7,2º * 50 = 360) i així generar el diagrama de radiació complert dels tags RFID, així com que aquest generi un fitxer de text on extreure les dades dels assajos.
- S'ha aconseguit implementar el botó de parada d'emergències "STOP", amb que els usuaris poder aturar el procés de mesura en qualsevol instant.
- S'ha aconseguit que la aplicació interfície pugui realitzar múltiples assajos consecutius sense la necessitat de reiniciar el sistema.
- S'ha validat el correcte funcionament del sistema a partir de la caracterització de diferents antenes RFID amb resultats plosius.

7.1 Continuació de treball

Si bé aquest projecte no observa treballs propis posteriors, en aquest apartat es realitzaran un conjunt de propostes de continuació de treball que poden permetre implementar noves funcions del sistema, així com possibles millores del mateix que es podrien implementar en les futures versions eventuals.

D'entrada, una de les principals funcionalitats que es podrien afegir serien les relacionades amb la introducció de nous assajos, els quals, a part de la potencia de detecció tinguin també presents altres variables com la freqüència utilitzada, el protocol de lectura o dades addicionals de les lectures.

Un altre aspecte que es podria implementar, seria la possibilitat de seleccionar el numero de mesures a realitzar, doncs si bé el sistema actual esta programat per a realitzar 50 mesures al llarg de la circumferència, els equips instal·lats permeten fins a 200 posicions. En aquesta mateixa línia, també es podrien implementar assajos de només una fracció de volta, per a ampliar les possibilitats d'un sistema que ara per ara només permet la mesura al llarg de tota la circumferència.

Finalment, una darrera implementació que es podria tenir en compte de cara a futures versions del sistema, seria la possibilitat de que la para d'emergència amb el botó "STOP" permetés la represa de la mesura, a diferencia del funcionament actual que requereix d'una reinicialització del procés de mesura.

7.2 Valoració Personal

Pel que refereix a la meva valoració personal respecta a aquest projecte, aquesta és àmpliament positiva, doncs a part de permetre'm introduir-me en un mon que amb prou feines coneixia, com és el de la radiofreqüència aplicada a la tecnologia RFID, he pogut explotar i incrementar les meves habilitats de programació. A part, també considero destacable el fet de que, gracies a un projecte no només centrat en el disseny del software sinó que també compta tasques de disseny mecànic, m'ha permès desenvolupar les meves habilitats en automatització d'una manera que no va ser possible en el TFG realitzat, on només vaig treballar la part del programari.

Addicionalment, també voldria destacar el fet de que, gràcies a aquest TFM, he pogut finalment aprendre i aprofundir en la programació de aplicacions amb el programa Visual Studio, el qual considero una gran eina que de ven segur em serà útil al llarg de la meva carrera professional.

Finalment, caldria assenyalar que l'únic element que lamento lleugerament és el fet de que el haver hagut de dedicar una fracció destacable del temps al disseny de la part mecànica del projecte (especialment en la llarga tasca de selecció del motor i del driver) van reduir la part de treball dedicada a la que és la meva veritable passió, la programació.

7.3 Agraïments

No voldria finalitzar la memòria d'aquest treball sense abans dedicar uns agraïments a totes les persones sense les que aquest projecte no hauria pogut ser realitat.

Primerament voldria agrair al director del projecte, Raúl Fernández, la feina realitzada al llarg d'aquest, sobretot per les múltiples reunions realitzades així com el contacte constant a traves de les converses per Mail.

D'igual manera, també voldria agrair als companys d'estudis Màster, especialment a en Jaume i a en Gerard, així com a la meva família, pel seu suport durant els mesos en que aquest projecte s'ha dut a terme, i sense qui aquest no hauria estat el mateix.

A tots, moltes gràcies.

8 Referències

[1] Colella, R., Catarinucci, L., & Tarricone, L. (2017). Measurement system for over-the-air evaluation of UHF RFID tags quality. Wireless Power Transfer, 4(1), 33-41. Cambridge University Press, 2016.

[2] RFID. A: Wikipedia [en línia]. Wikimedia Foundation, 2022. [Consulta: Juny 2022]. Disponible a: <https://en.wikipedia.org/wiki/Radio-frequency_identification>

[3] Stepper Motor. A: Wikipedia [en línia]. Wikimedia Foundation, 2022. [Consulta: Juny 2022]. Disponible a: <https://en.wikipedia.org/wiki/Stepper_motor>

[4] Interfacing stepper motor with PIC microcontroller. A: OpenLabPro [en línia]. Etiq Technologies, 2022 <https://openlabpro.com/guide/interfacing-stepper-motor-with-pic-microcontroller/>

[5] Microsoft Visual Studio. A: Wikipedia [en línia]. Wikimedia Foundation, 2022. [Consulta: Juny 2022]. Disponible a: <https://en.wikipedia.org/wiki/Microsoft_Visual_Studio>

[6] Visual Studio WinForm Windows Calculator Tutorial Example (C#). A: YouTube [en línia]. Google LLC, 2022. [Consulta: Març 2022]. Disponible a: <https://www.youtube.com/watch?v=Is1EHXFhEe4>

[7] About Us. A: Jadaktech [en línia]. Novanta Inc, 2022. [Consulta: Juny 2022]. Disponible a: <https://www.jadaktech.com/about-us/>

[8] ThingMagic UHF RFID Modules Development Kit. A: Jadaktech [en línia]. Novanta Inc, 2022. [Consulta: Juny 2022]. Disponible a: <https://www.jadaktech.com/products/thingmagic-rfid/thingmagic-uhf-rfid-modules-development-kit/>

[9] Mercury API Programmer's Guide. A: Jadaktech [en línia]. Novanta Inc, 2022. [Consulta: Juny 2022]. Disponible a: <https://www.jadaktech.com/resources/rfid-document-library/thingmagic-mercury-api-programmer-guide-v1-7/>

[10] M6e Hardware Guide. A: Jadaktech [en línia]. Novanta Inc, 2022. [Consulta: Juny 2022]. Disponible a: <https://www.jadaktech.com/resources/rfid-document-library/thingmagic-m6e-uhf-rain-rfid-module-family-hardware-guide-875-0053-09-revb/>

[11] Motor pas a pas RS PRO 535-0439. A: RS [en línia]. Amidata S.A.U, 2022. [Consulta: Juny 2022]. Disponible a: <https://es.rs-online.com/web/p/motores-paso-a-paso/5350439>

[12] Motor pas a pas RS PRO 180-5283. A: RS [en línia]. Amidata S.A.U, 2022. [Consulta: Juny 2022]. Disponible a: <https://es.rs-online.com/web/p/motores-paso-a-paso/1805283>

[13] Controlador de motor DC trifàsic Faulhaber. A: RS [en línia]. Amidata S.A.U, 2022. [Consulta: Juny 2022]. Disponible a: <https://es.rs-online.com/web/p/controladores-de-motores/9211331>

[14] Controlador de motor pas a pas RS PRO 905-8786. A: RS [en línia]. Amidata S.A.U, 2022. [Consulta: Juny 2022]. Disponible a: <https://es.rs-online.com/web/p/controladores-de-motores/9058786>

[15] Control Stepper Motor with L298N Motor Driver & Arduino. A: Last Minute Engineers [en línia]. LastMinuteEngineers.com, 2022. [Consulta: Juny 2022]. Disponible a: <https://lastminuteengineers.com/stepper-motor-l298n-arduino-tutorial/>

[16] H-bridge. A: Wikipedia [en línia]. Wikimedia Foundation, 2022. [Consulta: Juny 2022]. Disponible a: <https://en.wikipedia.org/wiki/H-bridge>

[17] Placa L298N. A: AMAZON [en línia]. Amazon.com, Inc, 2022. [Consulta: Juny 2022]. Disponible a: <https://www.amazon.es/Movilideas-Puente-Bridge-Stepper-Controlador/dp/B089DPPKQ1?th=1>

[18] Longruner Nema 17 Motor. A: AMAZON [en línia]. Amazon.com, Inc, 2022. [Consulta: Juny 2022]. Disponible a: <https://www.amazon.es/Longruner-Impresora-4-Cables-Conector-LD08/dp/B07FKH52S5?th=1>