Control automàtic de persianes elèctriques amb sensors de llum i pluja

Cristian Gascón Pérez
Enginyeria Informàtica – Itinerari Enginyeria de computadors
TFG - Arduino

Oriol Jaumandreu Sellarès
Pere Tuset Peiró
Gener 2017

Dedico aquest projecte a dues dones molt especials que m'han donat els ànims i suport necessari per enllestir aquest Treball Final de Grau, la meva mare i la meva dona.

FITXA DEL TREBALL FINAL

Títol del treball:	*Control automàtic de persianes elèctriques amb sensor de llum i pluja*
Nom de l'autor:	*Cristian Gascón Pérez*
Nom del consultor/a:	*Oriol Jaumandreu Sellarès*
Nom del PRA:	*Pere Tuset Peiró*
Data de lliurament (mm/aaaa):	*01/2017*
Titulació o programa:	*Pla d'estudis de l'estudiant*
Àrea del Treball Final:	*Arduino*
Idioma del treball:	*Català*
Paraules clau	*Arduino, Radio-freqüència, domòtica*

Abstract

The purpose of this project is to provide intelligent control blinds, which carried the movements according to the intensity of sunlight and the level of rain. It is designed with the Arduino electronic board that integrates a microcontroller and uses modules compatible with this device to ensure a smooth functioning of the whole.

The project is developed and validated in a real environment with two electric blinds in my home, these blinds have already built a RF remote control with a working frequency of 433.92 MHz. The device has the ability to get code commands the shutter (upload, download and save) prior to programming.

The design consists of four phases:

- Phase 1: Project Decision
- Phase 2: Setting and programming of all parties
- Phase 3: Programming set final system prototype.
- Phase 4: Functional tests and validation

Índex

Llista de figures

Llista de Taules

1 Introducció

Són molts els dispositius que interactuen amb el nostre món físic i prenen decisions intel·ligents en funció de diversos factors del medi físic, aquesta tecnologia té moltes aplicacions a la nostra llar o a processos industrials. Els últims avenços tecnològics de la maquinària i l'evolució de la programació fan cada vegada aquesta tasca més senzilla, tot això dóna pas a la computació física.

La computació física es centra en dissenyar dispositius que permeten establir un canal de comunicació entre el món físic i el món virtual de les màquines i ordinadors. Actualment són molts dispositius que posen en contacte aquest món físic amb el virtual i és una comunicació que es dóna en tots dos sentits.

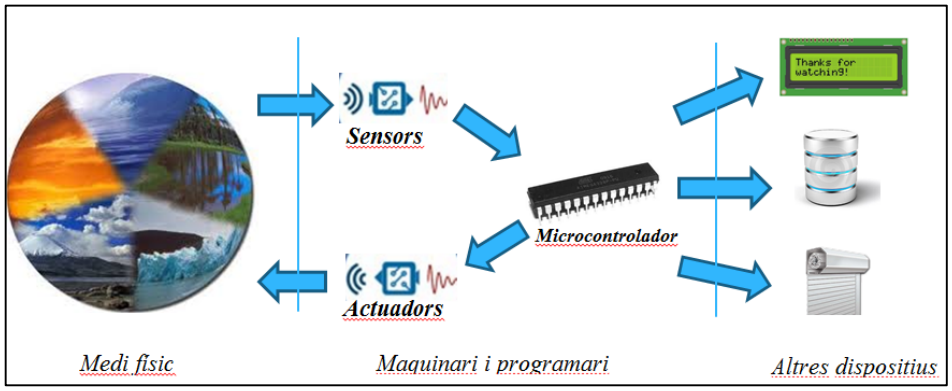


Figura 1 - Computació física

Aquesta comunicació amb el medi físic es fa gràcies a components com els sensors, que transformen els canvis d'energia produïts per les alteracions del medi físic en senyals elèctriques que un microcontrolador o qualsevol ordinador pot interpretar. En canvi, també existeix una comunicació amb el medi virtual mitjançant components com els actuadors, que transformen les senyals elèctriques en magnituds físiques.

Com a encarregats de la gestió dels sensors i actuadors tenim els microcontroladors, aquests fan els càlculs necessaris i prenen la decisió del que han de fer en tot moment segons una lògica programada; també es comuniquen amb altres dispositius per mostrar informació a l'usuari, emmagatzemar dades o, fins i tot, controlar altres dispositius.

1.1 Hipòtesi inicial

La finalitat d'aquest treball és dotar d'un control intel·ligent a unes persianes elèctriques, les quals realitzaran els moviments segons la intensitat de llum solar i el nivell de pluja; l'ordre de moviment es fa mitjançant una senyal de radio freqüència de 433.95Mhz.

El sistema disposa de sensors per mesurar les condicions ambientals, d'un mòdul de rellotge de temps real amb funció de calendari i d'una pantalla LCD on l'usuari pot obtenir informació addicional com la data, l' hora i l'estat de la persiana. Incorpora a més a més, mòduls de radio freqüència, un emissor que s'utilitza per la comunicació amb la persiana i un receptor utilitzat per la captació del codi de qualsevol comandament a distància. El disseny està pensat per funcionar amb qualsevol persiana elèctrica que ja tingui instal·lat un mòdul de control per RF.

El projecte està dissenyat amb elements *Open Hardware* basat en la plataforma *Arduino*, amb això, es pretén demostrar com aquesta plataforma es pot utilitzar per la comunicació del món virtual amb el món físic i fer que la computació física sigui més fàcil d'implementar. A més a més, es demostrarà com l' enginyeria inversa ens pot ser útil per estudiar el comportament d'un sistema existent i afegir noves funcionalitats a aquest mitjançant la plataforma *Arduino*.

1.2 Context i justificació del Treball

La realització d'aquest projecte ha estat motivat per la necessitat de controlar de manera automàtica el moviment d'unes persianes segons la intensitat de llum o el nivell de pluja, així com per fer un dispositiu versàtil, econòmic i que s'adapti a qualsevol persiana que disposi d'un control de ràdio freqüència.

Al mercat hi ha diversos dispositius comercials amb unes funcions específiques que ens aporta aquesta solució, però la majoria són molt costosos i requereixen d'una instal·lació addicional, a més a més, molt d'ells no són compatibles amb una persiana que ja disposi de control per ràdio freqüència.

Per aquests motius em va sorgir la idea de fer un sistema de control automàtic de persianes per ràdio freqüència amb *Arduino*, compatible amb qualsevol persiana controlada per RF i que aconsegueixi dotar d'un control intel·ligent el moviment de les persianes.

En definitiva, aquest sistema aportarà els següents avantatges:

- L'usuari no ha de controlar manualment les persianes quan plogui o faci molt de sol, ja que aquest sistema detectarà tant la pluja com la intensitat de la llum i donarà l'ordre automàtica de baixar o pujar les persianes segons convingui.

- L'ordre de moviment de les persianes es fa mitjançant RF, per tant s'evitarà fer instal·lacions addicionals per implementar el sistema.

- Pot controlar un número infinit de persianes i es compatible amb qualsevol sistema de persianes elèctriques amb un comandament a distància de ràdio freqüència de 433.92 MHz.

- És un sistema econòmic en comparació amb altres solucions comercials, el seu disseny amb Open Hardware fa que les despeses es redueixin.

- Possibilitat d'ampliar les funcions amb altres sensors o dispositius.

1.3 Objectius del Treball

L'objectiu principal d'aquest projecte és

- Dissenyar un sistema domòtic amb *Arduino* per controlar la pujada i baixada d'unes persianes elèctriques segons la intensitat de llum solar i de fenòmens meteorològics com la pluja.
- Compatibilitat amb qualsevol persiana elèctrica amb comandament a distància que funcioni amb una freqüència de 433.92 MHz.

Com a objectius secundaris tenim els següents:

- Mesura de temperatura i humitat de l'exterior
- Mesura d' intensitat de la llum
- Mesura de nivell de pluja
- Visualització per pantalla LCD d' informació a l'usuari com la data, hora, estat de la persiana, temperatura i humitat.
- Mantindrà la data i hora amb el dispositiu sense alimentació.

1.4 Enfocament i mètode seguit

Aquest sistema està dissenyat amb la placa electrònica *Arduino* que integra un microcontrolador i utilitza mòduls compatibles amb aquest dispositiu per garantir un bon funcionament del conjunt.

El projecte es desenvolupa i valida en un entorn real amb dues persianes elèctriques de la meva llar, aquestes persianes tenen ja incorporat un comandament a distància amb una freqüència de treball de 433.92 MHz. El dispositiu té la capacitat d'obtenir el codi de les ordres de la persiana (pujada, baixada i aturada), per l'obtenció d'aquest codi faré servir l' enginyeria inversa.

L'elaboració consta de 4 fases clarament diferenciades:

- Fase 1: Es pren la decisió del projecte, bàsicament es centra en la planificació del projecte, de la qual s'obté una estimació del temps i els recursos necessaris, a més a més, es fa un primer disseny del hardware i recerca d'informació.

- Fase 2: Durant aquesta fase s'adquireixen els components hardware i les llibreries necessàries pel funcionament, però el punt important d'aquesta fase és el muntatge i la programació de les diferents parts del sistema per obtenir un primer prototip.

- Fase 3: Es fa la programació de tot el conjunt i es genera el prototip final.

- Fase 4: Es centra en les proves funcionals i la validació de tot el sistema, té com a missió principal detectar possibles errades de disseny i millorar el funcionament.

1.5 Planificació del treballs

Per fer la planificació d'aquest projecte he tingut en compte les 5 dates clau corresponents al lliurament de les 3 PACs, la memòria final i la presentació. Per aconseguir aquestes fites he distribuït el projecte en 4 fases clarament diferenciades, les següents tables mostren cadascuna d'aquestes fases.

Etapa	Procès	Data Inici	Duració	Data finalització
PAC 1	Decisió del projecte i comunicació al consultor	22/9/16	1	23/9/16
	Planificació i realització diagrama de Gannt	24/9/16	3	27/9/16
	Recerca de components	28/9/16	2	30/9/16
	Disseny esquema connexió hardware	1/10/16	1	2/10/16
	Redacció i lliurament PAC 1	3/10/16	2	**5/10/16**

Taula 1 – Planificació fase 1

Etapa	Procès	Data Inici	Duració	Data finalització
PAC 2	Adquisició components hardware	6/10/16	4	10/10/16
	Instal·lació IDE Arduino	11/10/16	1	12/10/16
	Recerca de llibrerías Software per els components	13/10/16	5	18/10/16
	Muntatge, proves i programació LCD	19/10/16	3	22/10/16
	Muntatge, proves i programació LDR	23/10/16	3	26/10/16
	Muntatge, proves i programació Sensor de pluja	27/10/16	3	30/10/16
	Muntatge, proves i programació Sensor de temperatura i humitat	31/10/16	3	3/11/16
	Muntatge, proves i programació rellotge RTC	4/11/16	3	7/11/16
	Muntatge, proves i programació Receptor RF	8/11/16	3	11/11/16
	Captura codi RF del comandament a distància persianes	12/11/16	3	15/11/16
	Muntatge, proves i programació Emisor RF	16/11/16	3	19/11/16
	Redacció i lliurament PAC 2	20/11/16	3	**23/11/16**

Taula 2 – Planificació fase 2

Etapa	Procès	Data Inici	Duració	Data finalització
PAC 3	Programació conjunt sistema (Sensors, rellotge, LCD, emisor)	24/11/16	14	8/12/16
	Programació watchdog per resets automàtics	9/12/16	4	13/12/16
	Programació estalvi d'energia	14/12/16	4	18/12/16
	Redacció i lliurament PAC 3	19/12/16	2	**21/12/16**

Taula 3 – Planificació fase 3

Etapa	Procès	Data Inici	Duració	Data finalització
FINAL	Proves funcionament conjunt hardware i Software	22/12/16	5	27/12/16
	Validació	28/12/16	5	2/1/17
	Redacció i lliurament memòria final	3/1/17	12	**15/1/17**
	Realització Presentació i lliurament	16/1/17	6	**22/1/17**

Taula 4 – Planificació fase 4

A continuació tenim el diagrama de Gantt:

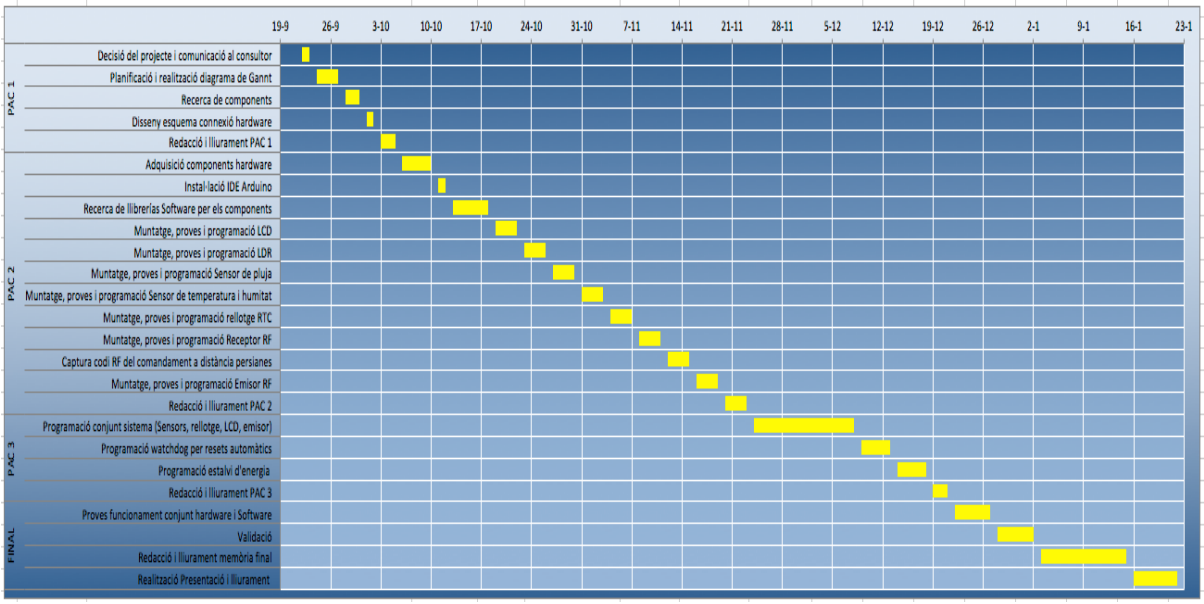


Figura 2 - Diagrama de Gannt

1.6 Productes obtinguts

En la realització d'aquest treball s'obté un dispositiu per controlar automàticament el moviment de les persianes mitjançant les condicions ambientals de l'exterior com la intensitat de llum o el nivell de pluja.

L'ordre de moviment de les persianes es fa mitjançant una senyal de ràdio freqüència, el disseny està preparat per obtenir el codi RF de funcionament de qualsevol persiana elèctrica, sempre que el seu comandament a distància tingui una freqüència de treball de 433.92Mhz.

El sistema disposa de 3 sensors ambientals (llum, pluja i temperatura) i d'una pantalla d'informació a l'usuari on es visualitzen dades com la data, hora, temperatura, humitat i la condició que ha fet moure la persiana.

A la imatge següent es mostra el producte obtingut amb els 3 sensors i la pantalla LCD d'informació a l'usuari.

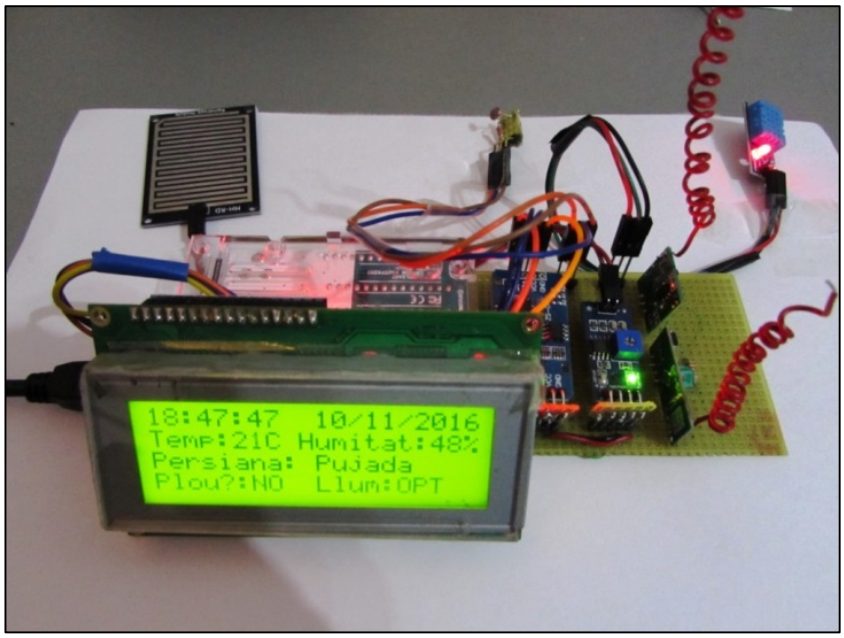


Figura 3 - Prototip

1.7 Breu descripció dels capítols de la memòria

Els capítols d'aquesta memòria estan distribuïts de la següent manera:

- Capítol 2. Antecedents. Descriu el funcionament actual de les persianes elèctriques i s'analitza una solució comercial del proveïdor Loxone.
- Capítol 3. Descripció funcional. Es fa una descripció funcional de tots els recursos necessaris i s'estableixen els requeriments del projecte.
- Capítol 4. Desenvolupament hardware. Descriu el disseny del maquinari.
- Capítol 5. Desenvolupament software. Descriu el disseny del programari
- Capítol 6. Proves funcionals i validació. Exposa totes les proves fetes per comprovar el funcionament descrit al capítol 3.
- Capítol 7. Instal·lació i configuració. Instruccions detallada de la instal·lació del sistema i la configuració del codi del comandament a distància i el calendari.
- Capítol 8. Valoració econòmica. Es fa el pressupost del projecte i es compara amb una solució comercial aportada per el proveïdor Loxone.
- Capítol 9. Propostes de millora. Es descriu les possibles millores del projecte a curt o llarg termini.
- Capítol 10. Conclusions. Apartat amb les conclusions generals del projecte i descripció d'unes propostes de millores del disseny.

2 Antecedents

2.1 Persianes elèctriques

Actualment totes les finestres de casa nostra disposen de persianes a totes les habitacions, per facilitar la pujada i baixada de les mateixes, es pot instal·lar un motor elèctric controlat per un comandament a distància per ràdio freqüència. Amb aquesta solució la pujada i baixada es fa de manera senzilla amb la pulsació d'un botó, però el control no és intel·ligent ja que l'ordre del moviment de la persiana el dóna el propi usuari, això presenta els següents inconvenients:

- Quan plou haig de baixar les persianes per evitar que les finestres s'embrutin, si no estic a casa, no puc baixar-les.
- Durant les vacances les persianes estan sempre baixades i això fa intuir que no hi ha ningú a casa amb la possible conseqüència d'un robatori.

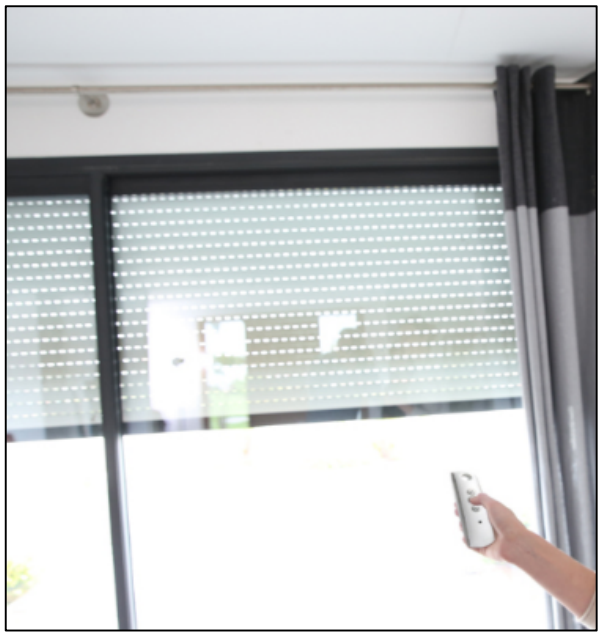


Figura 4 - Persiana amb control RF

2.2 Solució comercial

Actualment existeixen moltes solucions comercials per dotar d'un control intel·ligent a les persianes de la nostra llar, un exemple d'aquestes solucions la trobem en el proveïdor Loxone que fabrica tota mena de dispositius per la llar intel·ligent. Aquesta solució comercial mou automàticament la persiana en funció de la posició del sol, la pluja i, fins i tot, la temperatura objectiu que es vol dins la habitació.

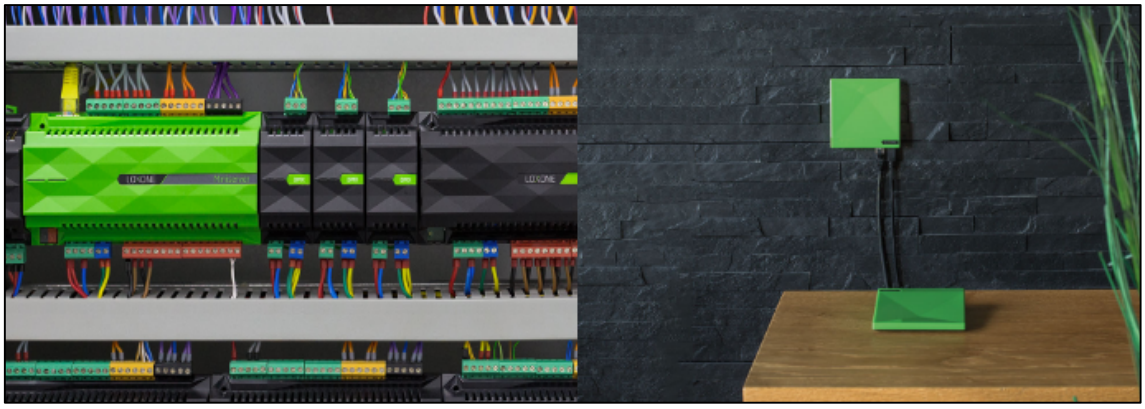


Figura 5 - Instal·lació Loxone

Aquest sistema requereix d'una instal·lació i configuració prèvia, també disposa d'una versió sense fils però només es compatible amb components del mateix proveïdor. Aquesta incompatibilitat ens obliga a haver d' adaptar els motors de les persianes i ens incrementa el cost de la solució.

3 Descripció funcional

3.1 Diagrama de blocs

Aquest sistema dissenyat consta de diversos elements físics, entre ells està *Arduino*, que és el principal component i el que porta tota la lògica de funcionament.

Com s'indica a la següent figura, el dispositiu *Arduino* permet la connexió i comunicació amb tots els elements; aquesta comunicació és bidireccional en el cas dels mòduls de ràdio freqüència i unidireccional amb la resta de mòduls.

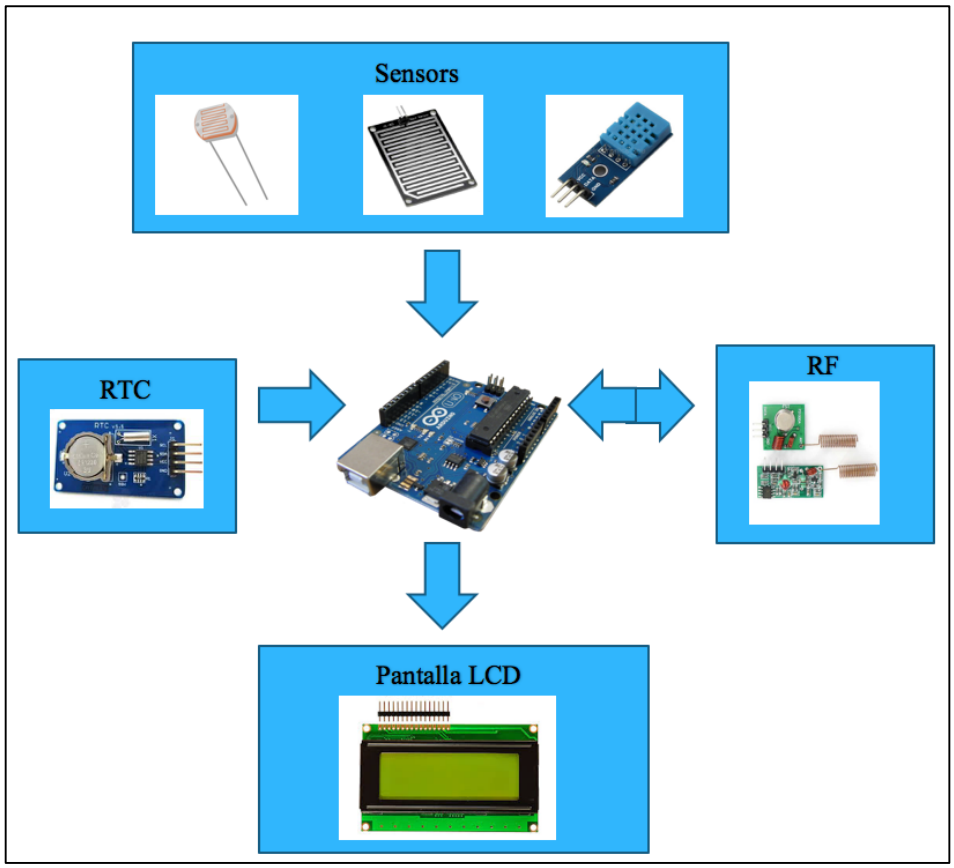


Figura 6 - Diagrama de blocs

3.2 Components físics

3.2.1 Arduino

El model escollit pel desenvolupament d'aquest projecte és *Arduino UNO*, incorpora un microcontrolador basat en el *ATmega 328* i conté tot el necessari pel seu funcionament. Les seves característiques són:

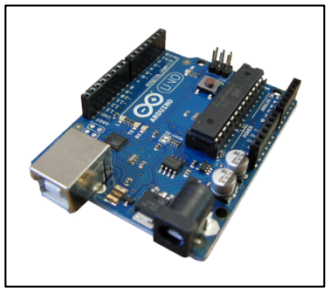


Figura 7 - Arduino UNO

- 14 Entrades/Sortides (6 amb PWM)
- 6 Entrades analògiques
- Bus i2C
- Alimentació entre 6-20V
- Velocitat del processador 16 MHz
- Capacitat de la memòria RAM de 8Kb

El dispositiu dissenyat només utilitza 1 entrada analògica, 3 entrades digitals, 1 sortida digital i el bus I2C.

3.2.2 Sensor de llum

El sensor de llum té com a finalitat mesurar la intensitat de llum solar i dóna l'ordre de moviment de la persiana segons les següents condicions establertes:

- Punt de llum màxim, aquest estat el dóna el sensor quan tenim llum solar directa, sempre donarà l'ordre de baixada de la persiana.
- Punt de llum òptim, aquest punt està entre l'alba i el capvespre però sense donar la llum solar directa, en aquest estat es dóna l'ordre de pujada de la persiana.
- Punt de llum nul, el dóna el sensor quan es de nit i donarà l'ordre de baixada de la persiana.

El sensor de llum consta d'una LDR, el qual és una resistència que varia el seu valor en funció de la intensitat de llum que rep. Les principals característiques són:

- Amb poca intensitat de llum té una resistència elevada.
- Amb molta intensitat de llum té una resistència baixa.

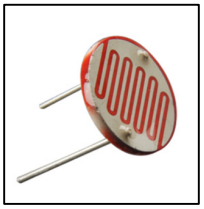


Figura 8 - Sensor LDR

3.2.3 Sensor de pluja

El sensor de pluja detecta el nivell de pluja i dóna l'ordre de moviment de les persianes segons aquestes dues condicions:

- Quan plou dóna l'ordre de baixada de les persianes.
- Quan no plou dóna l'ordre de pujada de les persianes.

El model de sensor de pluja és el YL-83, està compost d'una placa de baquelita amb pistes de coure amb una separació molt petita i d'una placa de control. Amb l'aigua es genera un curtcircuit a les pistes i es genera un camí de baixa resistència entre les pistes de polaritat positiva i negativa,

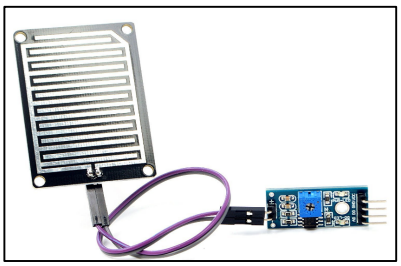


Figura 9 - Sensor pluja YL-83

- Tensió de funcionament 5V
- 2 sortides analògiques
- 1 sortida digital
- Control de sensibilitat

3.2.4 Sensor de temperatura i humitat

A més a més, el sistema disposa d'un sensor de temperatura i humitat el qual s'utilitza per prendre valors de temperatura i humitat relativa de l'ambient.

El DHT11 és el model escollit, està compost d'un sensor d'humitat capacitiu i un termistor, inclou un circuit integrat que realitza la conversió d'analògic a digital per enviar les dades a l'Arduino. Les principals característiques són les següents:

- Tensió de funcionament 5V
- Mesura temperatura de 0ºC a 50ºC
- Rang d'humitat del 20% al 95%

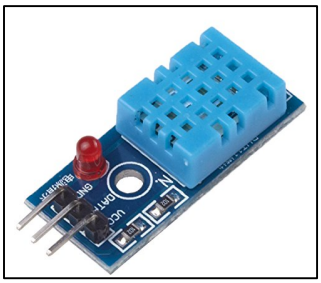


Figura 10 - Sensor DHT11

3.2.5 Pantalla LCD

El disseny incorpora una pantalla d'informació a l'usuari la qual dóna la següent informació a temps real:

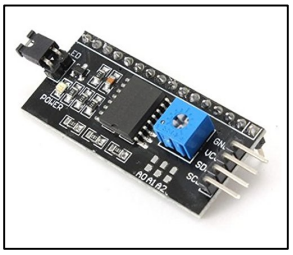
- Data i hora. Visualitza l' hora en format 24 hores i la data en format nacional.
- Temperatura i humitat. L'usuari pot consultar la temperatura en graus Celsius i la humitat relativa en percentatge, es visualitza el valor sense cap decimal.
- Estat de la persiana. Et dóna l'estat actual en el qual està la persiana, baixada, pujada o en moviment.
- Condició pluja. Tenim informació de la pluja.
- Condició llum. Et dóna informació de la llum captada (màxim, òptim i nul)

- Tensió de funcionament de 5V
- 20 caràcters per línia
- 4 línia de dades
- Sense bus I2C

Figura 11 - Pantalla LCD 20x4

3.2.6 Adaptador Bus I2C

La LCD 20x4 que utilitzarem no porta connexió I2C, a més l'Arduino escollit té un número limitat d'entrades/sortides, la solució està en instal·lar aquest mòdul i connectar la LCD en el bus I2C; així estalviem connexions a més de disposar d'un potenciòmetre per la regulació de la llum i contrast de la LCD.



- Tensió de funcionament de 5V
- Compatible amb pantalles LCD 16x2 i 20x4
- Adreça I2C entre 0x20 – 0x27

Figura 12 - Adaptador bus I2C

3.2.7 Mòduls de Radio Freqüència

Consta de 2 mòduls, el mòdul RF receptor que l'utilitzarem per llegir el codi del comandament a distància de la persiana i el mòdul RF transmissor, que l'utilitzarem per transmetre l' ordre de moviment de les persianes. A continuació tenim els mòduls RF amb les seves característiques:

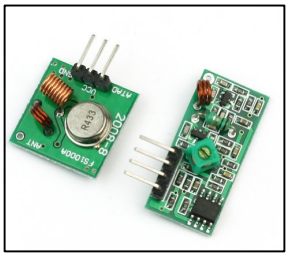


Figura 13 - Mòduls RF

- Tensió de funcionament de 5V
- Freqüència de treball de 315MHz – 433.92 MHz

3.2.8 Rellotge de temps real

El dispositiu incorpora un rellotge de temps real o RTC amb l'objectiu de dotar-ho d'un rellotge intern amb funcions de calendari, incorpora una bateria per garantir que en cas de perdre l'alimentació el sistema guardi la seva data i hora. Aquest sistema va connectat al bus I2C.

- Tensió de funcionament de 3.3V o 5V
- Bus I2C de 400kHz
- Funció de rellotge i calendari

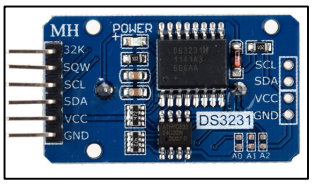


Figura 14 - Mòdul RTC DS3231

3.3 Components Software

3.3.1 Llibreries

Per fer funcionar tots els mòduls i sensors, s'utilitzen unes llibreries que simplifiquen molt la programació d'aquests i donem noves funcionalitats al nostre dispositiu, a la taula següent es mostren totes les llibreries utilitzades en aquest projecte.

Llibreries	Descripció
RCSwitch	Aquesta llibreria s'utilitza per a codificar i decodificar el senyal dels mòduls RF de 433Mhz.
DHT11	Aquesta llibreria s'utilitza per fer funcionar el sensor de temperatura i humitat DHT11
Time-master	Llibreria per a que Arduino tingui la possibilitat de mesurar el temps real, per exemple hora, dia, any, etc
DS1307_DS3231	Llibreria per utilitzar el mòdul de rellotge de temps real RTC
LiquidCrytal_I2C	Llibreria per utilitzar la pantalla LCD mitjançant el bus I2C

Taula 5 – Llibreries

3.3.2 Programari utilitzat

Per l' elaboració i desenvolupament del projecte he utilitzat programari compatible amb MacOsX, a la taula següent es mostra els programes utilitzats.

Programari	Descripció
Arduino IDE 1.6.12	Per la programació i compilació d'Arduino he utilitzat el IDE oficial.
Fritzing Versió 0.9.3 BETA	Programa per dissenyar els esquemes hardware del projecte.
Microsoft Excel 2016 per MAC	Paquet ofimàtic de fulla de càlcul per fer el diagrama de Gannt.
MicrosoftWord 2016 per MAC	Paquet ofimàtic de processador de text per la confecció de la memòria final
KeyNote Versió 7.0.5	Programa per confeccionar la presentació final de projecte
iMovie Versió 10.1.3	Programa d'edició de vídeo per confeccionar la presentació virtual del projecte
Text Edit Versió 1.11	Editor de text per visualitzar les llibreries d'Arduino i fer les modificacions adients en el codi.
I2C_scan	Programa que es carrega al Arduino per detectar els components del bus I2C i que ens doni la seva adreça dins el bus.

Taula 6 - Programari desenvolupament

3.4 Descripció senyal radio freqüència persianes

Actualment molts dispositius de la nostra llar, estan controlats per comandaments a distància que emeten senyals de ràdio freqüència. Funcionen amb una freqüència ultra ràpida (UHF) de 333Mhz, 433Mhz o fins hi tot de 833Mhz; tenen una longitud d'ona entre 1m i 100m.

En el cas de les persianes elèctriques, aquestes van dotades amb un dispositiu de RF amb comandament a distància, la seva freqüència de treball és de 433.92Mhz. A la imatge següent es mostra el comandament a distància amb les seves principals característiques.

- 1 bateria de 3V
- Freqüència d'emissió de 433.90Mhz
- 3 botons

Figura 15 - Comandament a distància

Aquest comandament a distància emet el codi mitjançant un senyal amb una freqüència variable, té les següents característiques:

- L' emissió comença amb uns bits de sincronització, durant uns 4500us emet un nivell alt i durant uns 1500us emet un nivell baix.

- Després d'aquest bits de sincronització, comença la transmissió de la dada i ho fa amb un senyal curt que té una durada de 250us i altre senyal més llarg amb una durada de 750us.

- El senyal està format per 20 bits dels quals els 16 primers serà idèntic per tots els botons del mateix comandament a distància i els 4 últims bits corresponen a l'ordre que volem enviar.

A les figures següents es veu un exemple del senyal que envia el comandament a distància de les persianes elèctriques, amb cadascun de les ordres (pujar, baixar i aturada).

Figura 16 - Senyal RF de pujada

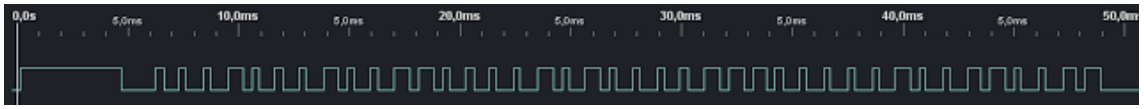


Figura 17 - Senyal RF de baixada

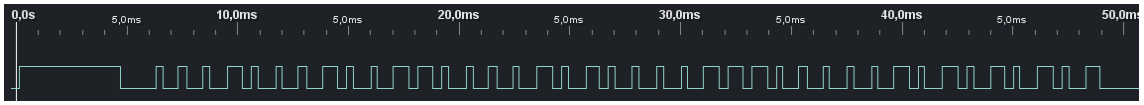


Figura 18 - Senyal RF aturada

4 Desenvolupament hardware

Una part molt important d'aquest projecte és el hardware i és per on es comença el seu desenvolupament. Com ja hem comentat en apartats anteriors, pel disseny s'utilitza la placa *Arduino UNO rev3* i mòduls compatibles *Open Hardware*. A la figura següent està representada la placa amb totes les senyals que disposa.

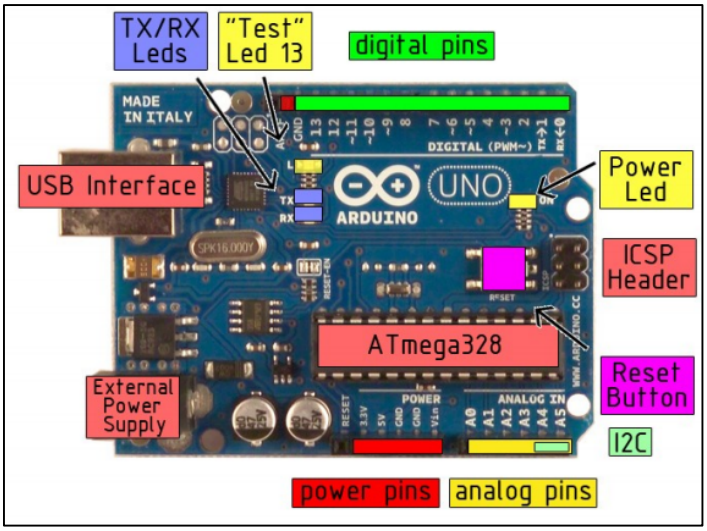


Figura 19 -Pins Arduino One

De totes les senyals només s'utilitzen 3 entrades digitals, 1 sortida digital, una entrada analògica i el bus I2C. La configuració és la següent:

- Entrades digitals: Receptor RF, sensor de pluja i sensor d'humitat/Temperatura
- Sortida digital: Emissor RF
- Entrada analògica: Sensor de llum LDR
- Sortida analògica: Cap
- Bus I2C: LCD i RTC

Tot el disseny està muntat en una placa de prototip i funciona alimentat amb un transformador que té una sortida de 5V i pot donar fins a 2A.

4.1 Senyals digitals

Una senyal digital és una variació de tensió entre 0 i la tensió d'alimentació però que no passa per valors intermedis, és a dir, només passa per dos estats que anomenem valor lògic *LOW* o "0" i valor lògic *HIGH* o "1". *Arduino One* disposa de 14 senyals digitals que són els pins del 0 al 13, aquestes senyals es poden configurar com a entrada o com a sortida.

Figura 20 - Pins digitals

De tots els senyals digitals només s'utilitzen un total de 4 per connectar els mòduls RF i els sensors de pluja i temperatura. A la següent tabla es detalla la configuració:

Mòdul	Entrada/sortida	Pin
Receptor RF	Entrada	Pin 2 (Interrupció)
Emissor RF	Sortida	Pin 9
Sensor Temperatura / humitat	Entrada	Pin 8
Sensor de pluja	Entrada	Pin 10

Taula 7- Configuració senyals digitals

El receptor RF està connectat en el pin 2 d'*Arduino* configurada com a entrada, aquesta entrada està com a interrupció per donar prioritat a la recepció del codi. Per evitar que aquesta interrupció s'activi pel soroll elèctric que podria captar el receptor RF, es posa una resistència de *pull-down* de 10K connectada a massa. El emissor RF va connectat en el pin 9 configurat com a sortida sense cap resistència *pull-down*.

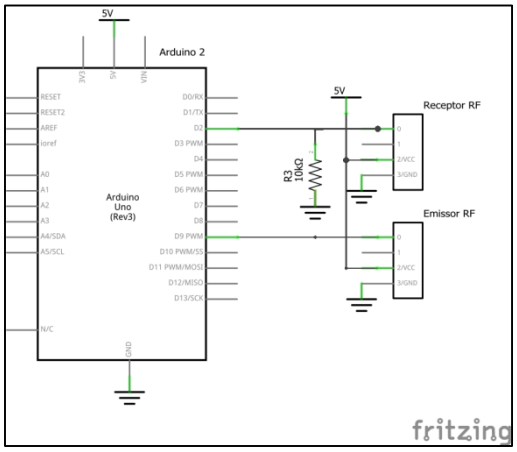


Figura 21 - Connexió mòduls RF

El sensor de pluja, temperatura i humitat van connectats en els pins 10 i 8 respectivament, tots dos sensors porten internament una resistència de *pull-down* per evitar sorolls per tant no es necessari posar aquesta resistència.

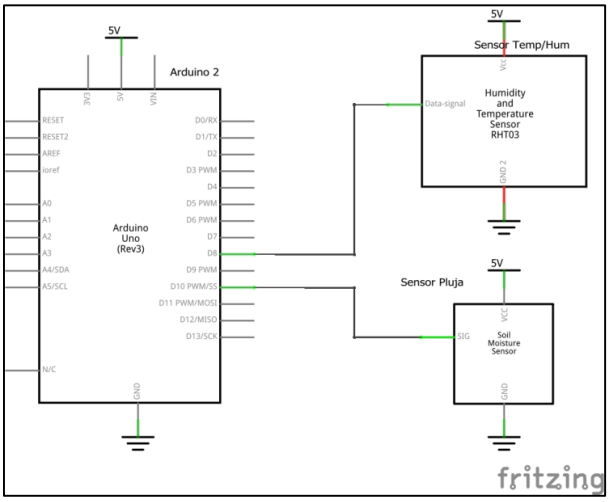


Figura 22 - Connexió sensors pluja, temperatura i humitat

4.2 Senyals analògics

Un senyal analògic pot prendre qualsevol valor dins d'un interval entre 0 i la tensió d'alimentació, en el nostre cas 5v. Arduino One disposa de 6 senyals digitals que van des del pin A0 fins A5, es poden configurar com a entrada o com a sortida.

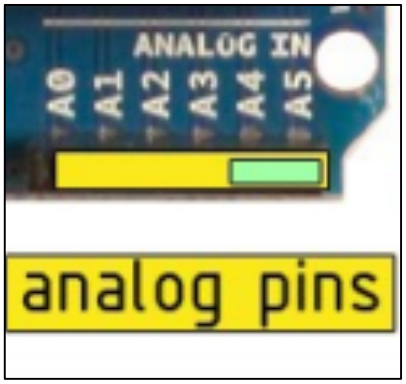


Figura 23 - Pins analògics

En el nostre disseny només s'utilitza la entrada A0 per connectar el sensor de llum, aquest consta d'un divisor de tensió format amb una LDR i una resistència de 220Ω. La resistència de la LDR varia sensiblement amb la quantitat de llum, amb molta llum la resistència és molt petita i amb poca llum la resistència és molt gran. Amb aquesta variació s'obté a la entrada A0 una tensió entre aproximadament 1.5V i 5V.

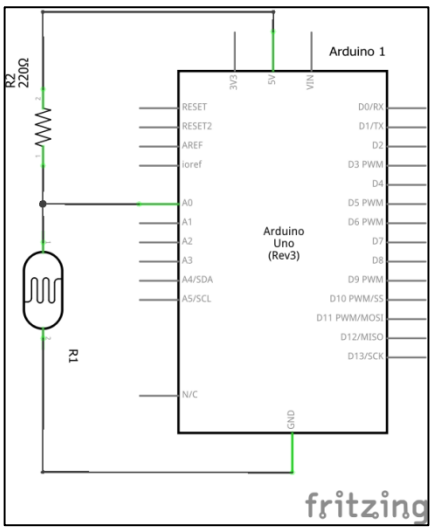


Figura 24 - Connexió sensor de llum

4.3 Bus I2C

El bus *I2C* (Inte-Integrated Circuit) és un bus de comunicacions molt utilitzat per la comunicació entre el microcontrolador i perifèrics com els sensors, l' estructura del bus de dades està formada per tres línies, una línia de dades (SDA), una segona línia amb un senyal de rellotge per la sincronització (SCL) i una última línia de massa que serveix de referència de les altres dues.

En el protocol *I2C* les línies de dades (SDA) i de rellotge (SCL) tenen una resistència Pull-up a la alimentació, aquesta resistència no cal afegir-la en el nostre disseny ja que *Arduino* incorpora aquestes resistències internament.

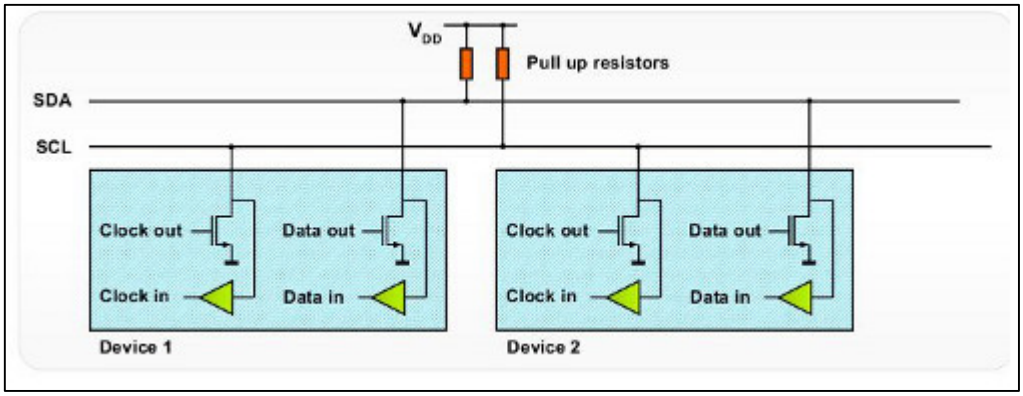


Figura 25 - Bus I2C

Aquest bus té una arquitectura de tipus Master-Slave, inicialment el dispositiu mestre inicia la comunicació amb els esclaus i pot enviar o rebre dades d'aquests, en canvi els esclaus no poden iniciar aquesta comunicació.

El bus I2C és síncron, el mestre proporciona un senyal de rellotge amb l'objectiu de mantenir sincronitzats a tots els dispositius connectats al bus. Això ens aporta els següents avantatges:

- No cal que cada dispositiu tingui el seu propi rellotge
- No fa falta negociar una velocitat de transmissió
- No són necessaris mecanismes per mantenir la transmissió sincronitzada

Arduino UNO té un bus I2C en els pins analògics A4 (SDA) i A5 (SCL), en aquest pins és on es connecta el rellotge RTC i la LCD.

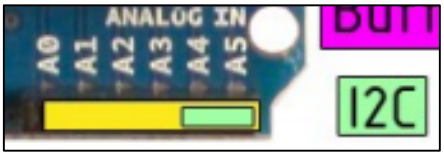


Figura 26 - Pins I2C

En el mòdul del rellotge RTC la connexió al bus I2C es fa directament, en canvi, en el cas de la pantalla LCD cal d'un adaptador bus I2C. Aquest adaptador del bus I2C es compatible amb LCD's de 16x2, 16x4 o 20x4 i ja porta incorporat una resistència variable per canviar la intensitat de lluminositat de la LCD, amb aquesta adaptació es redueix considerablement les connexions i facilita el funcionament de la LCD.

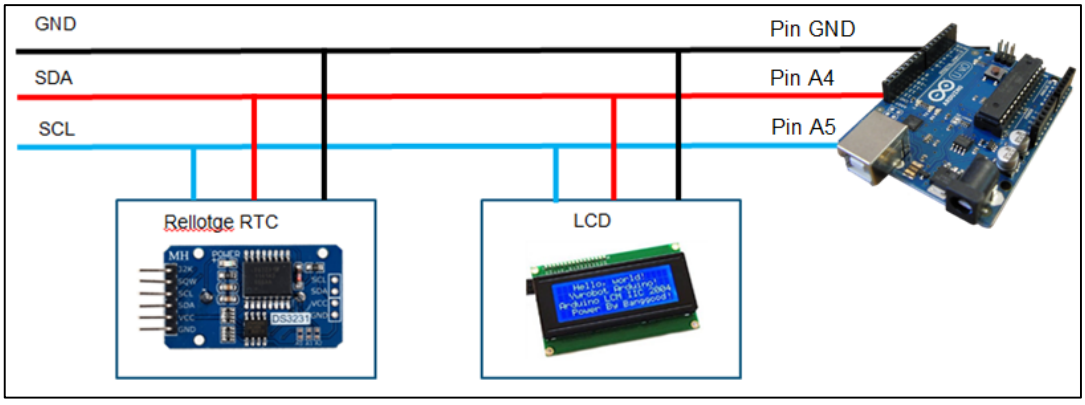


Figura 27 - Connexió RTC i LCD amb el bus I2C

5 Desenvolupament Software

Una vegada muntat el *hardware* es realitza la programació de cada part per separat, d'aquesta manera s'eviten problemes d'implementació. El programari del sistema ha estat desenvolupat amb la *IDE Arduino 1.6.12*, consta de varies parts diferenciades:

- **Declaració variables i llibreries**: Es fa una declaració de totes les variables i les llibreries que utilitzarem en tot el programa.
- **Configuració inicial**: Aquestes instruccions només s'executen quan encenem per primera vegada *Arduino* o quan fem un reset, inclou la configuració de la hora, el missatge inicial del sistema, la inicialització dels sensors, la configuració dels mòduls RF i la inicialització de la comunicació sèrie per la captura del codi.
- **Loop**: Aquesta és la part principal del programa que s'executa de manera seqüencial en un bucle continuo.
- **Funcions**: Per garantir una programació estructurada, s'ha dividit el programa en funcions, aquesta manera facilita molt la tasca de desenvolupament del programari.
- **Interrupció**: Aquesta senyal interromp la activitat normal del programa per atendre un procés més prioritari, que en el nostre cas s'utilitza per la captació del codi del comandament a distància.

5.1 Implementació sensors

El valor analògic del sensor de llum es llegeix directament de la entrada analògica A0 i es fa una conversió a digital per obtenir un valor entre 0 i 1024, el valor queda emmagatzemats en la variable *ldr*, Aquest valor anirà variant segons la intensitat de llum.

Per acomplir el requeriments d'activació de l'ordre de moviment de les persianes segons les condicions de llum, s'estableix 3 llindars: *ldrUmbralMax*, l*drUmbralOptim* i *ldrUmbralNul.* Amb aquests llindars i el valor llegit de la variable *ldr* es fa la comprovació per determinar el nivell de llum que tenim (màxim, òptim o nul). A les figures següents es mostra la implementació dels llindars amb els seus valors i com es fa la comprovació del nivell de llum.

```
37  int ldrUmbralMax = 120;   // Umbral Punt de llum màxim,
38  int ldrUmbralOptim = 600; // Umbral Punt de llum òptim,
39  int ldrUmbralNul = 1000;  // Umbral Punt de llum nul, e
```

Figura 28 - Codi llindars de nivell de llum

```
115   //Comprovació estat de llum per determinar l'estat Maxim, Optim i nul
116
117   if (ldr > ldrUmbralNul) {
118       llumMax = false;
119       llumOptim = false;
120       llumNul = true;
121     }
122
123     if (ldr < ldrUmbralOptim) {
124       llumMax = false;
125       llumOptim = true;
126       llumNul = false;
127     }
128
129     if (ldr < ldrUmbralMax){
130       llumMax = true;
131       llumOptim = false;
132       llumNul = false;
133     }
```

Figura 29 - Codi comprovació nivell de llum

El valor digital del sensor de pluja es llegeix directament de la entrada digital 10 i no fa falta cap llibreria, és una senyal digital que ens indicarà amb un 0 si plou i amb un 1 si no plou. La detecció es fa mitjançant una comparació lògica amb la variable anomenada *pluja*.

La dada del sensor de temperatura i humitat es llegeix de la entrada digital 8 però fa falta la llibreria DHT11 per obtenir el seu valor; aquesta llibreria ja incorpora totes les funcions necessàries per el funcionament d'aquest sensor. Per aconseguir els valors d'aquestes dues magnituds físiques es fa una lectura mitjançant la trucada a les funcions *readHumidity()* i *readTemperature()* i s'emmagatzema el valor de la humitat en la variable *h* i el valor de la temperatura en la variable *t*. A la figura següent es mostra la part del codi que fa aquesta funció.

```
103
104   int h = dht.readHumidity(); //Es fa lectura del valor de humitat i es po
105   int t = dht.readTemperature(); //Es fa lectura del valor de humitat i es
106
```

Figura 30 - Codi lectura temperatura i humitat

5.2 Implementació moviment persianes

L'activació del moviment de les persianes consta de dues funcions, per la pujada de la persiana s'utilitza la funció *emisorPujar()* i per la baixada la funció *emisorBaixar()*. Per determinar quan pujar o baixar la persiana, prèviament es verifica si hi ha pluja i quin nivell de llum tenim, el codi d'aquesta condició el tenim a la figura següent.

```
137   if ((llumNul) || (llumMax) || (pluja == 0)) {
138       emisorBaixar();
139     } else if (llumOptim) {
140       emisorPujar();
141     }
142 }
```

Figura 31 - Codi comparació LDR i sensor pluja

Totes dues funcions fan una comprovació prèvia de l'estat de la persiana i continuen l'execució sempre i quan la persiana estigui en l'estat contrari a l'ordre que s'envia, es a dir, si la persiana està pujada podrem enviar l'ordre de baixada però no podrem enviar aquest ordre si la persiana està baixada.

Mitjançant la instància *mySwitch.sendQuadState (codi)* s'envia el codi de RF per pujar o baixar cadascuna de les persianes, una vegada enviat el codi es canvia el valor de les variables que indica l'estat de la persiana.

```
149 //Codi del emisor per pujar persiana
150 void emisorPujar(){
151
152   if (persianaBaixada) {         //Si la persiana està baixada activem la pujada
153
154     lcd.setCursor( 10, 2 );        // ir a la quarta linia
155     lcd.print("Pujant...");
156
157     mySwitch.sendQuadState (codiPujar1); //Enviem el codi RF per pujar la persiana 2
158     delay (200);
159     mySwitch.sendQuadState (codiPujar2); //Enviem el codi RF per pujar la persiana 2
160
161     delay(5000);  //Donem un temps per baixar la persiana
162
163     persianaPujada = true;
164     persianaBaixada = false;
165   }
166 }
167
168
169 //Codigo del emisor para baixar la persiana
170 void emisorBaixar(){
171
172   if (persianaPujada) {        //Si la persiana està pujada activem la baixada
173     lcd.setCursor( 10, 2 );        // ir a la quarta linia
174     lcd.print("Baixant...");
175
176     mySwitch.sendQuadState (codiBaixar1); //Enviem el codi RF per baixar la persiana 1
177     delay(200);
178     mySwitch.sendQuadState (codiBaixar2); //Enviem el codi RF per baixar la persiana 2
179
180     delay(5000); //Donem un temps per baixar la persiana
181
182     persianaPujada = false;
183     persianaBaixada = true;
184   }
185 }
```

Figura 32 - Codi moviment persianes

5.3 Implementació LCD

La programació de la pantalla LCD es compon de varies funcions, cadascuna de elles serveix per visualitzar diferents dades:

- *missatgeInicialLcd()* : Aquesta funció s'utilitza per configurar els paràmetres de la pantalla LCD com la lluminositat i el tipus de caràcters que pot contenir, en aquest cas és una LCD de 20 caràcters i 4 línies. A més a més presenta per pantalla el missatge de benvinguda quan s'encén el dispositiu.
- *visualitzacióHoraRTC()*: S'utilitza per visualitzar la data i la hora, aquestes dades les obté del rellotge de temps real mitjançant una instància de *rtc.getData()*
- *visualitzacióMeteoLcd()*: S'utilitza per visualitzar la temperatura i la humitat que dona el sensor, agafa les dades de dues variables *temp* i *hum.*
- *visualitzacióLlumLcd()*: S'utilitza per presentar per la pantalla LCD l'estat dels 3 nivells de llum (màxim, òptim, null), sempre agafarà la variable que estigui amb valor cert.
- *visualitzacióPlujaLcd()*: Presenta per la pantalla si plou, llegeix directament la variable del sensor de pluja, si aquesta pren valor 0 significa que plou i si pren valor 1 significa que no plou.
- *lcdEstatPersiana()*: Presenta per pantalla l'estat de la persiana, llegeix el valor de la variable *persianaPujada* si aquesta es certa ens indica que la persiana estarà pujada, pel contrari si està en fals ens indica que la persiana està baixada.

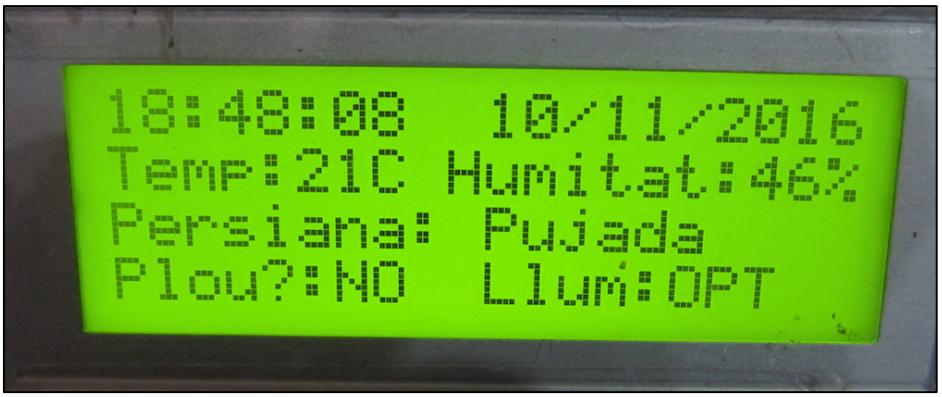


Figura 33 - LCD amb tota la informació

5.4 Implementació rellotge RTC

Per implementar la funcionalitat de temps, s'utilitzen les llibreries *Time.h* i *RTC.h*, aquestes incorporen funcions per donar la data i la hora en temps real. Les funcions que es necessita són les següents:

- *rtc.getData()*: Aquesta funció s'utilitza per obtenir la hora i la data real del mòdul hardware RTC. El valor s'emmagatzema en la variable d.
- *rtc.setDateTime ()*: Aquesta funció s'utilitza per posar la data i hora real al dispositiu, com a paràmetres s'ha de posar en número l'any, mes, dia, hora, minuts i segons.

5.5 Implementació captació codi

Per garantir la màxima prioritat durant la captura del codi del comandament a distància i que es pugui capturar el codi sense perdre cap dada, es fa mitjançant una interrupció. Aquesta interrupció s'inicialitza durant la configuració inicial i es fa en conjunt amb la inicialització de la comunicació sèrie.

```
//Inicialització comunicació serie per la captura del codi comandament
Serial.begin(9600);
Serial.println("Preparat...");
Serial.println("Polsi un botó del comandament");
pinMode(2, INPUT);    //Activo el pin 2 com a entrada en Arduino ONE aqu
attachInterrupt(0, capturaInterrupcio, CHANGE);  //Aquesta interrupció s
```

Figura 34 – Codi inicialització captura codi comandament

La interrupció es fa amb la entrada 0 i s'activarà sempre que es detecti un canvi de senyal, l'activació fa una trucada a la funció *capturaInterrupció(),* aquesta s'encarrega de llegir la senyal de radio freqüència del comandament, la descodifica en bits i envia per el port sèrie el codi.

6 Proves funcionals i validació

El projecte es desenvolupa i valida en un entorn real amb dues persianes elèctriques que ja tenen incorporat un comandament a distància amb una freqüència de treball de 433.92 MHz. Per determinar si el projecte compleix els requeriments funcionals descrits en el capítol 3 es fan les següents proves:

- **Captació de codi comandament a distància**. Es prova la captació de codi de dos comandaments a distància diferents i la captació es correcta en els tres polsadors (pujada, baixada i aturada)

- **Activació moviment persiana mitjançant sensor de pluja**. Es posa aigua en el sensor de temperatura, es verifica que el dispositiu detecta pluja i envia l'ordre de baixada a les persianes.

- **Activació moviment persiana mitjançant sensor de llum LDR**. Es prova els tres nivells de llum:
    - Nivell de llum òptim: amb llum ambient es verifica en la pantalla que tenim aquest nivell, amb aquest es dóna ordre de pujada a les persianes.
    - Nivell de llum màxim: s'aplica llum directa al sensor LDR i es verifica que en la pantalla apareix nivell màxim, amb aquest nivell es dóna ordre de baixada de les persianes.
    - Nivell de llum nul: es tapa el sensor de llum i es verifica que en la pantalla apareix nivell nul, amb aquest nivell es dóna ordre de baixada de les persianes.

- **Proves sensor de temperatura i humitat**. Es verifica en la pantalla LCD que es visualitza temperatura i humitat, es fa pujar la temperatura i es comprova com la dada canvia. S'aplica una mica d'humitat al sensor i es comprova que la dada varia.

7 Instal·lació i configuració

7.1 Instal·lació

El dispositiu requereix d'una configuració inicial que consisteix bàsicament en compilar el programa amb les llibreries utilitzades i fer la càrrega d'aquesta programació al dispositiu. Els passos són els següents

- Connectem el dispositiu al port USB del nostre ordinador.
- Afegim les llibreries necessàries a l'IDE Arduino, des de el menú '*Programa – Incloure llibreria – Afegir llibreria .ZIP'*

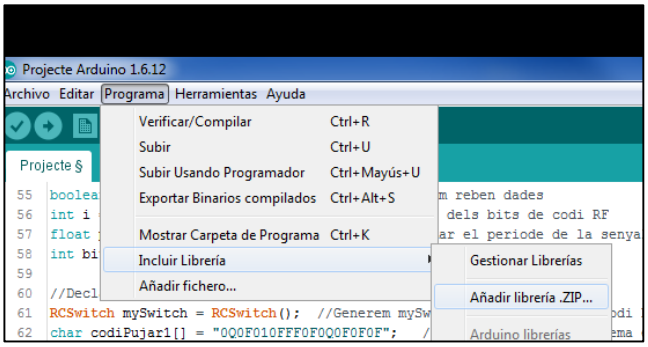


Figura 35 - Afegir llibreries Arduino

- Obrim l'arxiu amb la programació per el control automàtic de persianes i el compilem, no ha de donar errades.
- Transferim el codi al dispositiu per mitjà del botó "Pujar".

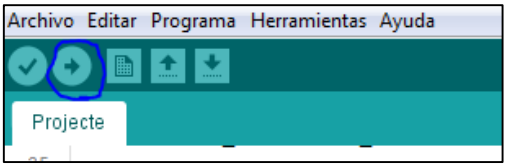


Figura 36 - Transferir codi

- Connectem el dispositiu la alimentació de xarxa, sortirà primer un missatge de benvinguda durant 5 segons, a continuació apareixerà la pantalla d'informació a l'usuari.

7.2 Configuració data i hora

El dispositiu disposa d'un rellotge RTC amb una pila per emmagatzemar l´hora, però en cas que s'hagi de posar en hora s'ha de seguir els següents passos:

- Connectem el dispositiu al port USB del nostre ordinador, sortirà primer un missatge de benvinguda durant 5 segons, a continuació apareixerà la pantalla d'informació a l'usuari.

- Obrim la programació del projecte amb el IDE d'Arduino i localitzem la línia de configuració inicial `rtc.setDateTime`

```
//Configuració inicial
void setup() {

  //rtc.setDateTime( 2016, 10, 31, 15, 05, 00 );
```

Figura 37 - Configuració data i hora inactiva

- Trèiem el comentari ("//") d'aquesta línia i actualitzem la data i hora, els dígits de esquerra a dreta representa el següent: any, mes, dia, hora, minuts i segons.

```
//Configuració inicial
void setup() {

  rtc.setDateTime( 2016, 10, 31, 15, 05, 00 );
```

Figura 38 - Configuració data i hora activa

- Transferim el codi al dispositiu per mitjà del botó "Pujar".

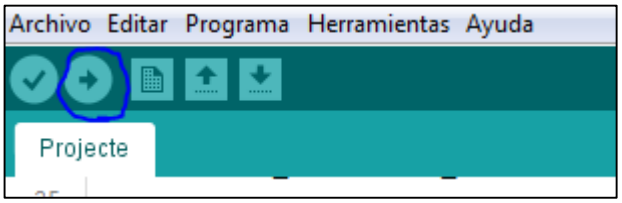


Figura 39 - Transferir codi

- Per últim tornem a posar el comentari a la línia `rtc.setDateTime` i transferim de nou el codi al dispositiu.

7.3 Configuració codi comandament a distància

Abans d'utilitzar el producte es necessari realitzar la configuració del codi del comandament a distància de les nostres persianes, els passos a seguir són els següents:

- Connectem el dispositiu al port USB del nostre ordinador, sortirà primer un missatge de benvinguda durant 5 segons, a continuació apareixerà la pantalla d'informació a l'usuari.

- Iniciem el IDE d'Arduino i executem el monitor sèrie, ens sortirà una pantalla per llegir el codi del comandament a distància

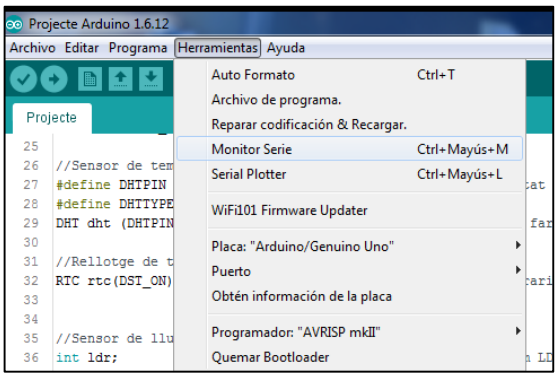


Figura 40 - Monitor serie IDE Arduino

- Polsem qualsevol botó del comandament a distància i per pantalla ens sortirà el codi, el codi sortirà 4 vegades per cada pulsació de botó. S'ha de capturar el codi dels 3 botons del comandament a distància.

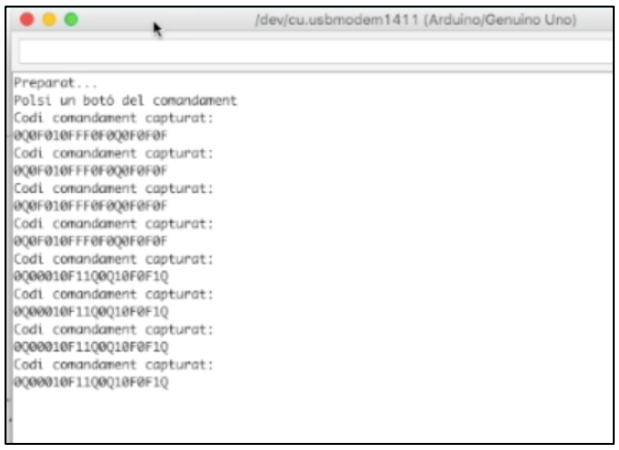


Figura 41 – Captura de codi

- Copiem el codi capturat i l'introduïm a l'apartat de declaració variables.

```
//Declarem variables per l'envio del codi del comandament
RCSwitch mySwitch = RCSwitch();  //Generem mySwitch per l'
char codiPujar1[] = "0Q0F010FFF0F0Q0F0F0F";   //Constant q
char codiBaixar1[] = "0Q0F010FFF0F0Q0F0110";  //Constant q
char codiStop1[] = "0Q0F010FFF0F0Q0FFFFF";    // Constant

char codiPujar2[] = "0Q00010F11Q0Q10F0F1Q";   //Constant q
char codiBaixar2[] = "0Q00010F11Q0Q10F0110";  //Constant q
char codiStop2[] = "0Q00010F11Q0Q10FFFFF";    // Constant
```

Figura 42 - Codi declaració variables codi comandament

- Transferim el codi al dispositiu per mitjà del botó "subir".

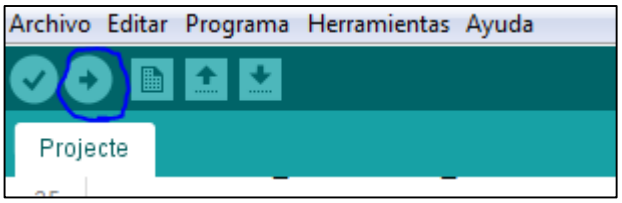


Figura 43 - Transferir Codi

8 Valoració econòmica

8.1 Pressupost projecte

Es realitza una valoració econòmica del treball realitzat, he tingut en compte l'estudi, materials, desenvolupament, instal·lació i posada en marxa.

	Unitats / Hores	Preu unitat	Import
Estudi			
Definició conceptual projecte	2	30,00 €	60,00 €
Definició objectius projecte	2	30,00 €	60,00 €
Materials			
Placa Arduino Uno rev 3	1	21,33 €	21,33 €
Mòdul LCD 20x4	1	4,35 €	4,35 €
Mòdul 433Mhz RF Receptor i Transmissor	1	3,99 €	3,99 €
Sensor de pluja YL-83	1	8,90 €	8,90 €
Mòdul Interface I2C per LCD 20x4	1	3,80 €	3,80 €
Mòdul RTC DS3231	1	1,60 €	1,60 €
Sensor temperatura i humitat DHT 11	1	2,60 €	2,60 €
Resistència 10K	1	0,05 €	0,05 €
Placa de nodes PCB	1	4,80 €	4,80 €
Cables	1	2,79 €	2,79 €
Desenvolupament			
Desenvolupament codi	20	30,00 €	600,00 €
Muntatge en placa PCB	8	30,00 €	240,00 €
Proves validació	5	30,00 €	150,00 €
Instal·lació i posada en marxa			
Instal·lació d'equip	2	20,00 €	40,00 €
Posada en hora rellotge RTC	1	20,00 €	20,00 €
Captació codi comandament a distància	1	20,00 €	20,00 €
Proves funcionals	2	20,00 €	40,00 €
TOTAL			1.284,21 €

Taula 8 - Valoració econòmica projecte

A la gràfica següent es pot observar com el 74% de les despeses corresponent al desenvolupament del dispositiu, en canvi solament un 5% correspon als materials per la seva construcció. Aquesta reducció de despeses en els materials es gràcies al fet d'utilitzar components Open Hardware i que l'usuari pot utilitzar les seves persianes elèctriques.

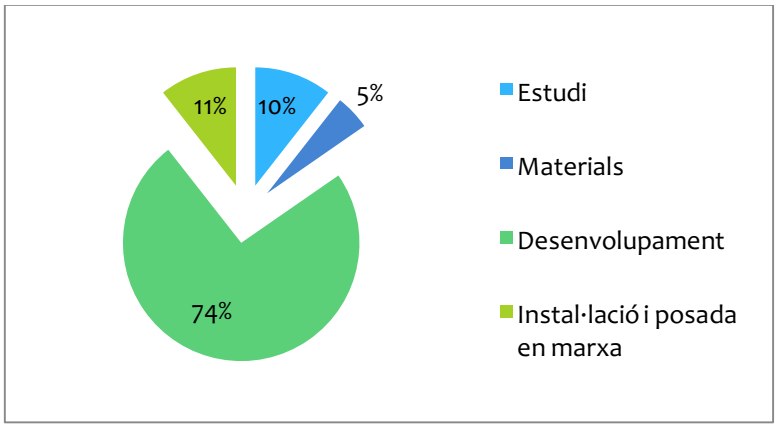


Figura 44 - Gràfica pressupost projecte

8.2 Pressupost producte comercial

En el mercat existeixen dispositius comercials capaços de dotar un control intel·ligent a les persianes, un proveïdor que ens aporta una solució similar a la que es vol implementar amb aquest projecte es Loxone. A continuació es detalla el pressupost d'aquesta solució, s'ha tingut en compte el preu dels materials i la instal·lació.

	Unitats / Hores	Preu unitat	Import
Materials			
Sensor de luminositat 0-10V	1	96,68 €	96,68 €
Sensor de pluja 24VDC	1	70,06 €	70,06 €
Miniserver (Centralita de control)	1	502,15 €	502,15 €
Motor persiana SolidLine Air	2	199,65 €	399,30 €
Sensor de temperatura i humitat Exterior	1	141,45 €	141,45 €
Air Base Extension	1	99,83 €	99,83 €
Font alimentació	1	32,55 €	32,55 €
Instal·lació i posada en marxa			
Instal·lació d'equip	5	25,00 €	125,00 €
TOTAL			1.467,02 €

Taula 9 - Valoració econòmica Loxone

A la gràfica següent es pot observar com el 91% de les despeses correspon al material i només un 9% a la instal·lació.

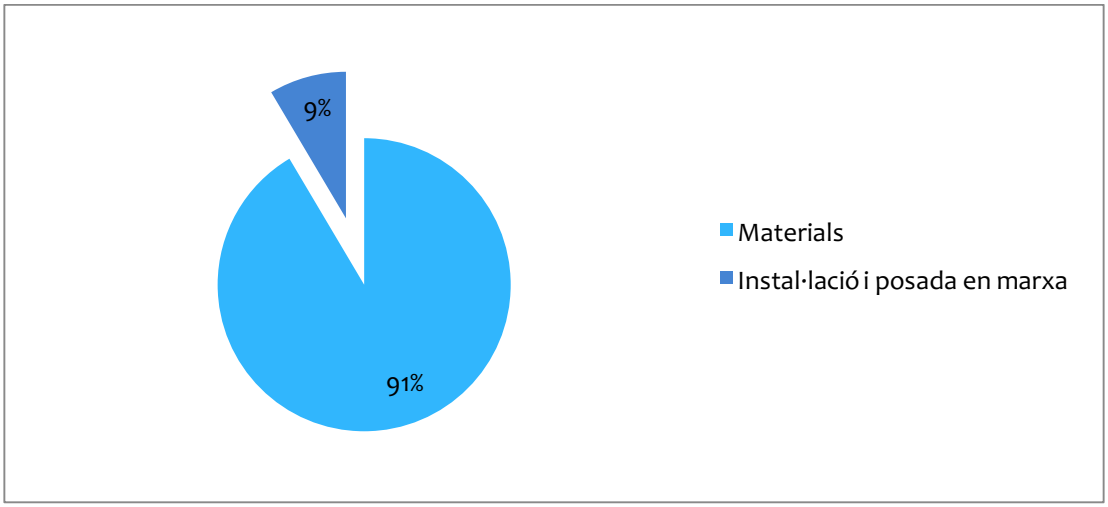


Figura 45 - Gràfica comparació instal·lació i material

8.3 Comparació

El nostre projecte en comparació amb l'opció comercial Loxone és un 25% més econòmica, de fet el preu incrementa per les hores de desenvolupament.

	Projecte	Loxone
Estudi	120,00 €	- €
Materials	54,21 €	1.342,02 €
Desenvolupament	840,00 €	- €
Instal·lació i posada en marxa	120,00 €	125,00 €
TOTAL	1.134,21 €	1.467,02 €

Taula 10 - Comparació econòmica projecte i Loxone

A la gràfica següent es pot observar com el cost dels materials és molt superior en l'opció Loxone, aquest estalvi dels materials es degut a l'ús de components Open Hardware que són més econòmics.

Figura 46 - Comparació pressupost

9 Propostes de millora

Aquest projecte s'ha desenvolupat amb *Arduino UNO*, aquest serveix perfectament com a prototip però degut a la seva mida dificulta la integració del dispositiu dins la caixa de les persianes elèctriques. Per millorar aquesta integració es preveu fer un nou disseny amb la nova placa *d'Arduino MKR1000*, la qual te unes mides molt reduïdes.

Tota la informació a l'usuari la dona la pantalla LCD, aquesta informació es podria consultar des de qualsevol dispositiu mòbil, per això es preveu aprofitar les característiques del *wifi* integrat del dispositiu *MKR1000 i* dissenyar un servidor web amb tota aquesta informació, amb aquesta millora es podria prescindir de la pantalla LCD i consultar-la des de Internet.

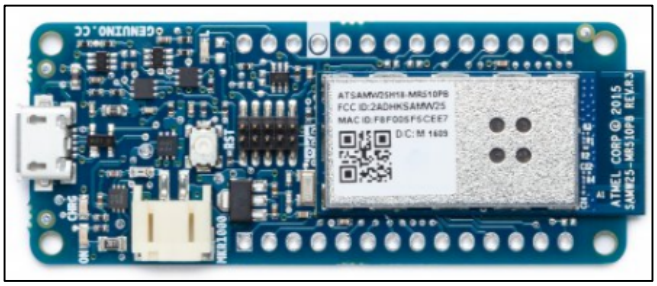


Figura 47 - Placa Arduino MKR1000

El control de les persianes es totalment automàtic i no té cap funcionalitat de moviments manuals per voluntat pròpia del usuari, per això es s' implementarà una nova funcionalitat de funcionament manual. Amb aquesta nova funció i amb el servidor web, es podria controlar la persiana còmodament amb el mòbil.

El dispositiu disposa d'un rellotge de temps real que només s'utilitza per visualitzar la hora i la data actuals, aquest RTC té funcions d'alarmes que es poden aprofitar per programació d'horaris de moviment de les persianes.

10 Conclusions

Una vegada finalitzat el projecte es el moment d'analitzar i treure les conclusions adients, en els següents paràgrafs detallo aquestes conclusions.

Tots els objectius proposats en el projecte han estat assolits amb èxit, i el resultat final té el comportament esperat segons la hipòtesis inicial. S'ha dissenyat un sistema domòtic per controlar de manera intel·ligent el moviment d'unes persianes elèctriques, les quals realitzen els moviments segons la intensitat de llum solar i el nivell de pluja; l'ordre de moviment es fa mitjançant una senyal de radio freqüència de 433.95Mhz i es totalment compatible amb qualsevol persiana que ja tingui instal·lat un mòdul de radiofreqüència.

El resultat de totes les proves funcionals i de validació realitzades en el prototip són satisfactòries i el dispositiu funciona segons els requeriments establerts, no obstant això, té una limitació alhora de la seva implantació en un escenari real, per exemple els sensors s'han d'instal·lar a la intempèrie i no compleixen els requeriments d'estanqueïtat, això podria fer malbé els components electrònics i que el funcionament no sigui el correcte.

Amb aquest projecte he demostrat com l'ús del Open Hardware facilita una comunicació entre el món virtual i el món físic. A més s'ha demostrat com la enginyeria inversa ens ha sigut útil per estudiar el funcionament del comandament a distància de les persianes i aconseguir afegir noves funcionalitats per dotar a la persiana d'un control intel·ligent, similar a altres solucions comercials.

En quan a la part econòmica destacar que el meu projecte és aproximadament un 25% més econòmic que qualsevol altra solució comercial, del cost total només un 5% és dels components, si això el comparem amb la solució comercial *Loxone* es pot concloure que he aconseguit un model més econòmic.

La realització d'aquest projecte ha suposat un repte personal, el prototip compleix totes les meves expectatives, a més a més ha sigut una tasca molt enriquidora i un pas per introduir-me en l'apassionant món de *Arduino*.

11 Glossari

Arduino: És una placa de circuit imprès basada en un microcontrolador de codi obert.

I2C: És un bus de comunicació sèrie, el seu nom ve d'Inter-Integrated Circuit i es compon de 3 línies: SDA (dades), SCL (rellotge) i GND (massa)

LDR: Component electrònic que la seva resistència disminueix amb l'augment d'intensitat de la llum, també s'anomena fotoresistor, fotoconductor, etc. Les seves sigles en anglès *light-dependent resistor*

LCD: De l'anglès *Liquid Crystal Display* o pantalla de cristall líquid.

RF: Radio Freqüència

RTC : De l'anglès *Real Time Clock*, és el rellotge de temps real.

Interrupció: és una senyal rebuda per el microprocessador que indica l'aturada de tots els processos per executar un altra més prioritari.

Open Hardware: maquinari amb els esquemes d'accés públic i es poden aconseguir amb un baix cost.

Open Software: programació de codi obert

12 Bibliografia

1- http://www.prometec.net/ , [data consulta 29 de setembre del 2016]

2- http://fritzing.org/ , [data consulta 29 de setembre del 2016]

3- http://www.i2c-bus.org/ , [data consulta 30 de setembre del 2016]

4- http://robologs.net/tutoriales/arduino/ , [data consulta 1 d´octubre del 2016]

5- http://physudo-e.blogspot.com.es/2013/08/home-automation-with-arduino-and-433_16.html , [data consulta 15 d'Octubre del 2016]

6- http://blogingenieria.com/general/ingenieria-inversa/ [data consulta 25 d'Octubre del 2016]

7- http://www.loxone.com/eses/smart-home/funcionalidades/persianas.html [data consulta 2 de Novembre del 2016]

8- http://www.sublimexmotors.com/ [data consulta 2 de Novembre del 2016]

9- https://www.smarthome.com/ [data consulta 25 d'Octubre del 2016]

10- http://www.luisllamas.es/2016/05/arduino-i2c/ [data consulta 20 de Novembre del 2016]

11- http://tronixstuff.com/2010/10/20/tutorial-arduino-and-the-i2c-bus/ [data consulta 20 de Novembre del 2016]

Annex 1. Esquema elèctric

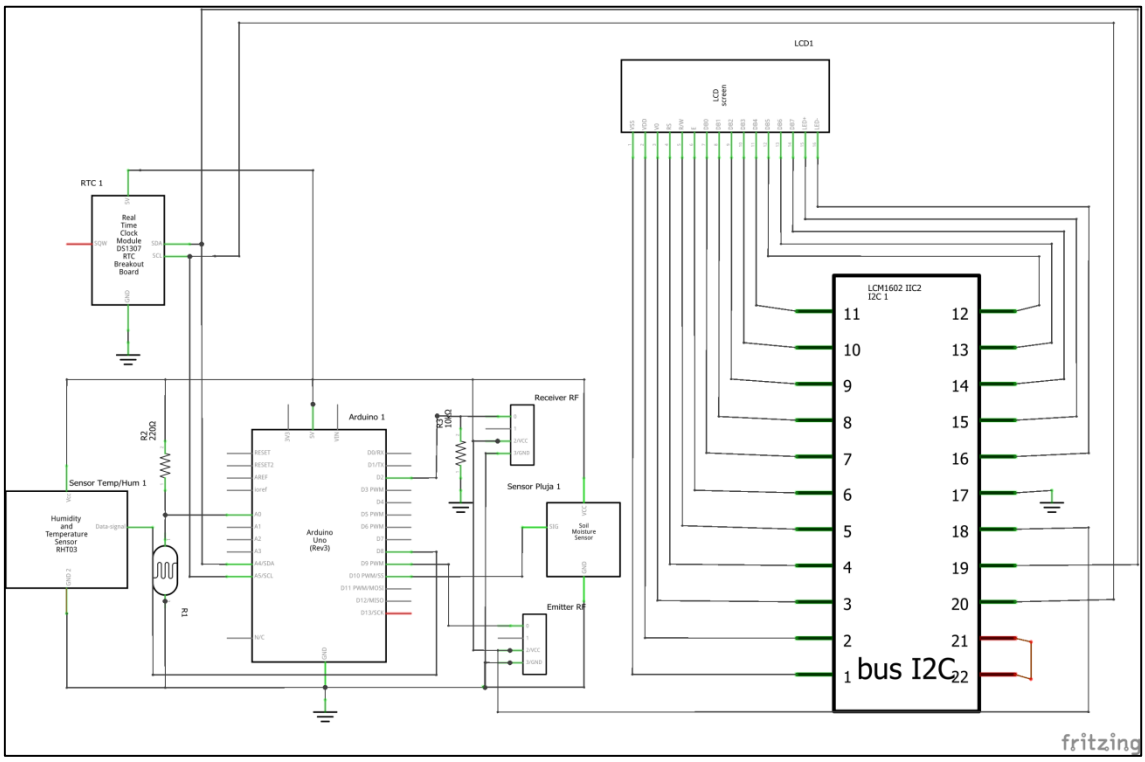


Figura 48 - Esquema elèctric

Annex 2. Programació Arduino

```
1
2
3  /*
4   * TFG - ARDUINO
5   * Grau Enginyeria Informatica
6   * Cristian Gascón Pérez
7   *
8   * Control automàtic de persianes elèctriques amb sensors de llum i pluja
9   */
10
11 #include <DHT.h>
12 #include <Time.h>
13 #include <TimeLib.h>
14 #include <Wire.h>
15 #include <LCD.h>
16 #include <LiquidCrystal_I2C.h>
17 #include <LiquidCrystal.h>
18 #include <RTC.h>
19 #include <RCSwitch.h>
20
21
22 //Pantalla LCD
23 #define I2C_ADDR  0x27  //Definim la variable amb la direcció del Display LCD
24 LiquidCrystal_I2C lcd(I2C_ADDR,2, 1, 0, 4, 5, 6, 7);  //Generem una instancia
25
26 //Sensor de temperatura
27 #define DHTPIN 8  //Definim la entrada digital on va connectat el sensor de t
28 #define DHTTYPE DHT11 //Definim el tipus de sensor DHT11
29 DHT dht (DHTPIN,DHTTYPE); //Es defineix la variable dht que farà la comunicac
30
31 //Rellotge de temps real (RTC)
32 RTC rtc(DST_ON); // Activem el canvi d'hora automátic en horari Estiu
33
34
35 //Sensor de llum LDR
36 int ldr;          //variable per el valor del sensor de llum LDR
37 int ldrUmbralMax = 120;   // Umbral Punt de llum màxim, aquest estat el dona
38 int ldrUmbralOptim = 600; // Umbral Punt de llum òptim, aquest punt està entr
39 int ldrUmbralNul = 1000;  // Umbral Punt de llum nul, el dona el sensor quan
40 int pluja;
41 boolean llumMax = false;  //Variable que ens indica que este a punt de llum m
42 boolean llumOptim = true; //Variable que ens indica que este a punt de llum ò
43 boolean llumNul = false;  //Variable que ens indica que este a punt de llum n
```

```
43  boolean llumNul = false;  //Variable que ens indica que este a punt de llum r
44
45  //Estat Persiana
46  boolean persianaBaixada = true;    //variable que ens indica si la persiana e
47  boolean persianaPujada = false;    //variable que ens indica si la persiana e
48
49  //Sensor pluja
50  int plujaIn=10;     // Entrada digital del sensor de pluja
51
52  //Declarem variables i vector per la captura del codi del comandament
53  boolean capturant = false;  //Ens indica si estem capturant dades
54  boolean comprovacioRF = false;   //Ens indica si s'ha capturat alguna senyal
55  boolean repDada = false;    //Ens indica si estem reben dades
56  int i = 0;  //Iniciem el contador per la captura dels bits de codi RF
57  float periodeRF = 0;   //Variable on emmmagatzemar el periode de la senyal
58  int bitsRF[40]; //Vector per emmagatzemar el codi de 40 bits
59
60  //Declarem variables per l'envio del codi del comandament
61  RCSwitch mySwitch = RCSwitch();  //Generem mySwitch per l'envio del codi RF
62  char codiPujar1[] = "0Q0F010FFF0F0Q0F0F0F";   //Constant que emmagatzema el c
63  char codiBaixar1[] = "0Q0F010FFF0F0Q0F0110";  //Constant que emmagatzema el c
64  char codiStop1[] = "0Q0F010FFF0F0Q0FFFFF";    // Constant que emmagatzema el
65
66  char codiPujar2[] = "0Q00010F11Q0Q10F0F1Q";   //Constant que emmagatzema el c
67  char codiBaixar2[] = "0Q00010F11Q0Q10F0110";  //Constant que emmagatzema el c
68  char codiStop2[] = "0Q00010F11Q0Q10FFFFF";    // Constant que emmagatzema el
69
70
71  //Configuració inicial
72  void setup() {
73
74    //rtc.setDateTime( 2016, 10, 31, 15, 05, 00 );  //Aquesta linia s'utilitza
75
76    missatgeInicialLcd ();  //Imprimim missatge inicial per pantalla LCD durant
77    delay (5000);
78
79    mySwitch.enableTransmit(9);  //Transmitim per el pin 9 de la placa arduino
80    mySwitch.setProtocol(4);    //Seleccionem protocol QUADSTATE
81
82    dht.begin();    //Iniciem el sensor de temperatura
```

```
83
84
85    //Inicialització comunicació serie per la captura del codi comandament
86    Serial.begin(9600);
87    Serial.println("Preparat...");
88    Serial.println("Polsi un botó del comandament");
89    pinMode(2, INPUT);    //Activo el pin 2 com a entrada en Arduino ONE aquest
90    attachInterrupt(0, capturaInterrupcio, CHANGE);  //Aquesta interrupció s'ac
91
92  }
93
94
95
96  //*****Codi Principal*****************************
97
98  void loop() {
99
100
101   ldr = analogRead (0);   //llegeix el valor analogic de la entrada A0 que ir
102   pluja = digitalRead (plujaIn);  //llegeix el valor digital de la entrada am
103
104   int h = dht.readHumidity(); //Es fa lectura del valor de humitat i es posa
105   int t = dht.readTemperature(); //Es fa lectura del valor de humitat i es pc
106
107
108   lcd.clear ();   // Fem un esborrat de la pantalla LCD i imprimim l'estat de
109   lcdEstatPersiana();
110   visualitzacioHoraRTC();
111   visualitzacioMeteoLcd(t,h); //Es pasen per parametres el valor de temperatu
112   visualitzacioPlujaLcd();
113   visualitzacioLlumLcd();
114
115   //Comprovació estat de llum per determinar l'estat Maxim, Optim i nul
116
117   if (ldr > ldrUmbralNul) {
118       llumMax = false;
119       llumOptim = false;
120       llumNul = true;
121     }
```

```
122
123     if (ldr < ldrUmbralOptim) {
124       llumMax = false;
125       llumOptim = true;
126       llumNul = false;
127     }
128
129     if (ldr < ldrUmbralMax){
130       llumMax = true;
131       llumOptim = false;
132       llumNul = false;
133     }
134
135   //Es compara l'estat de la LDR i del sensor de pluja per pujar o baixar la
136
137   if ((llumNul) || (llumMax) || (pluja == 0)) {
138       emisorBaixar();
139     } else if (llumOptim) {
140       emisorPujar();
141     }
142 }
143
144 //************************************************
145
146
147 //******Funcions programa**********************
148
149 //Codi del emisor per pujar persiana
150 void emisorPujar(){
151
152   if (persianaBaixada) {         //Si la persiana està baixada activem la puj
153
154     lcd.setCursor( 10, 2 );        // ir a la quarta linia
155     lcd.print("Pujant...");
156
157     mySwitch.sendQuadState (codiPujar1); //Enviem el codi RF per pujar la per
158     delay (200);
159     mySwitch.sendQuadState (codiPujar2); //Enviem el codi RF per pujar la per
160
161     delay(5000);  //Donem un temps per baixar la persiana
```

```
162
163     persianaPujada = true;
164     persianaBaixada = false;
165   }
166 }
167
168
169 //Codigo del emisor para baixar la persiana
170 void emisorBaixar(){
171
172   if (persianaPujada) {        //Si la persiana està pujada activem la baixad
173     lcd.setCursor( 10, 2 );        // ir a la quarta linia
174     lcd.print("Baixant...");
175
176     mySwitch.sendQuadState (codiBaixar1); //Enviem el codi RF per baixar la p
177     delay(200);
178     mySwitch.sendQuadState (codiBaixar2); //Enviem el codi RF per baixar la p
179
180     delay(5000); //Donem un temps per baixar la persiana
181
182     persianaPujada = false;
183     persianaBaixada = true;
184   }
185 }
186
187
188 //Codi per la visualització estat persiana per LCD
189 void lcdEstatPersiana (){
190
191   lcd.setCursor ( 0, 2 );        // ir a la tercera linia
192   lcd.print("Persiana:");
193
194   switch (persianaPujada){
195
196     case true:
197     lcd.setCursor( 10, 2 );        // ir a la quarta linia
198     lcd.print("Pujada");
199     break;
200
```

```
201     case false:
202     lcd.setCursor ( 10, 2 );        // ir a la quarta linia
203     lcd.print("Baixada");
204     break;
205   }
206 }
207
208
209 //Codi per la inicialització LCD
210 void missatgeInicialLcd () {
211
212   lcd.begin (20,4);    // Inicialtizar el display amb 20 caracters 4 linias
213   lcd.setBacklightPin(3,POSITIVE);
214   lcd.setBacklight(HIGH);
215
216   lcd.home ();
217   lcd.print("TFG - ARDUINO - UOC");
218   lcd.setCursor ( 2, 1 );        // ir a la segona linia
219   lcd.print("Control persianes");
220   lcd.setCursor ( 5, 2 );        // ir a la tercera linia
221   lcd.print("amb sensors");
222   lcd.setCursor ( 0, 3 );        // ir a la cuarta linia
223   lcd.print("- Cristian Gascon -");
224 }
225
226 //Codi per visualitzar la hora i la data per la LCD
227 void visualitzacioHoraRTC(){
228
229   Data d = rtc.getData();
230
231   lcd.setCursor (0,0);
232   lcd.print (d.toString((char*)"H:i:s"));
233   lcd.setCursor(10, 0);
234   lcd.print( d.toString((char*)"d/m/Y") );
235 }
236
237
238
```

```
238
239 // Codi per visualitzar temperatura i humitat per la LCD, rep com a parametre
240 void visualitzacioMeteoLcd(int temp, int hum) {
241   lcd.setCursor(0, 1);
242   lcd.print("Temp:");
243   lcd.print (temp);
244   lcd.print("C");
245   lcd.setCursor(9, 1);
246   lcd.print("Humitat:");
247   lcd.print (hum);
248   lcd.print("%");
249 }
250
251 // Codi per visualitzar estat llum.
252 void visualitzacioLlumLcd() {
253
254   lcd.setCursor(10, 3);
255   lcd.print("Llum:");
256   lcd.setCursor(15, 3);
257
258   if (llumMax){
259     lcd.print("MAX");
260   }
261
262   if (llumOptim){
263     lcd.print("OPT");
264   }
265
266   if (llumNul){
267     lcd.print("NUL");
268   }
269
270 }
271
272
273
274 // Codi per visualitzar si hi ha pluja o no.
275 void visualitzacioPlujaLcd() {
276
277   lcd.setCursor(0, 3);
278   lcd.print("Plou?:");
279   lcd.setCursor(6, 3);
280
```

```
281    switch (pluja) {
282
283      case 0:
284      lcd.print("SI");
285      break;
286
287      case 1:
288      lcd.print("NO");
289      break;
290    }
291  }
292
293
294  // Interrupció per la captura de codi
295
296  void capturaInterrupcio() {
297
298    if (!capturant) {  //Si no estem capturant codi
299      if (!comprovacioRF) {  // Si no hi ha comprovació de senyal RF
300        if (digitalRead(2) == HIGH) {  //Si hi ha canvi de nivell baix (LOW) a
301          periodeRF = micros();
302          comprovacioRF = true;
303        }
304      }
305
306      else {    //Si estem capturan codi
307
308        // Si la senyal a nivel alt (HIGH) té un periode més gran que 4000us i
309        if ((micros() - periodeRF > 4000) && (digitalRead(2) == LOW)) {
310          comprovacioRF = false;
311          capturant = true;
312          periodeRF = micros();
313        }
314
315        else {
316          // Si no hi ha senyal el posem a false
317          comprovacioRF = false;
318        }
319      }
320    }
321
322    else {  // Comencem a capturar el codi
323      if (!repDada) {  //Si no hem rebut dades
```

```
      if ((micros() - periodeRF > 1000) && digitalRead(2) == HIGH) {  //that
        repDada = true; //hem rebut dades
        periodeRF = micros();
      }
    }

    else {  //rebem les dades del codi
      //En el flanc de pujada (HIGH)
      if (digitalRead(2) == HIGH) {
        //Comprovem el periode de la senyal
        periodeRF = micros();
      }

      //En el flanc de baixada (OW)
      else if (digitalRead(2) == LOW) {
        // Comprovem el periode de la senyal
        if (micros() - periodeRF > 500) {
          //posem la dada 1 en el vector
          bitsRF[i] = 1;
        }

        else {
          //posem la dada 0 en el vector
          bitsRF[i] = 0;
        }

        if (i < 39) {
          // Comprovem que tenim tots els bits rebuts si no el tenim sumem co
          i++;
        }

        else {
          //Finalitzada la captura del codi
          noInterrupts();  //Posem la interrupció a OFF


          //Imprimint el codi amb el protocol "quad-bit" el llegim del vector

          Serial.println("Codi comandament capturat:");

          //Comencem recorregut per el vector per llegir els bits
          for (i = 0; i <= 38; i = i + 2) {
            // Si hi ha dos bits a 0, imprimim 0
```

```
            if ((bitsRF[i] == 0) && (bitsRF[i+1] == 0)){
              Serial.print("0");

            }
            // Si hi ha un bit a 0 i el bit següent a 1 , imprimim F
            else if ((bitsRF[i] == 0) && (bitsRF[i+1] == 1)){
              Serial.print("F");

            }
            // Si hi ha un bit a 1 i el bit següent a 0 , imprimim Q
            else if ((bitsRF[i] == 1) && (bitsRF[i+1] == 0)){
              Serial.print("Q");

            }
            // Si hi ha dos bits a 1, imprimim 1
            else if ((bitsRF[i] == 1) && (bitsRF[i+1] == 1)){
              Serial.print("1");

            }

          }
          Serial.println(); //Fem un salt de línia
          i = 0;  //iniciem el contador per la impresió del codi
          repDada = false;    //Posem les variables de recepció dades i capt
          capturant = false;
          interrupts();  //Posem la interrupció a ON
          return;  //Fem un return per començar de nou
        }
      }

    }
  }
}


//********************FI DE PROGRAMA*****************************************
```

Annex 3. Captures de pantalla prototip

Figura 49 - Pantalla presentació

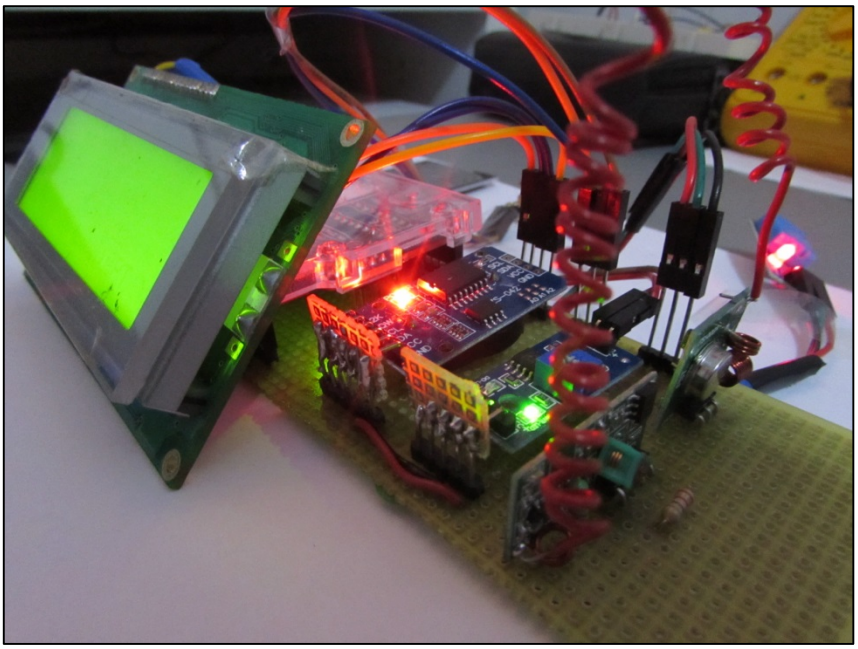


Figura 50 - Vista prototip

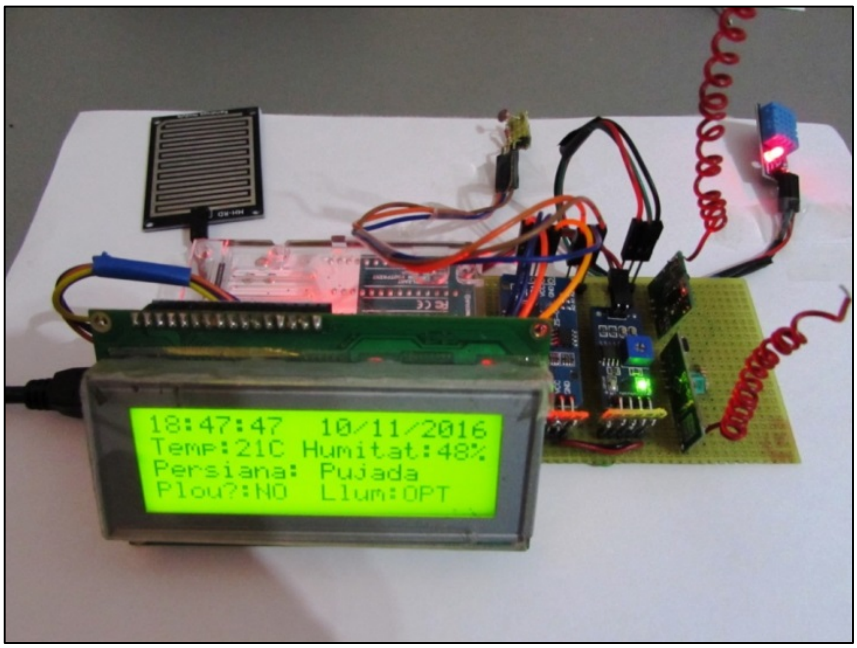


Figura 51 - Pantalla de dades

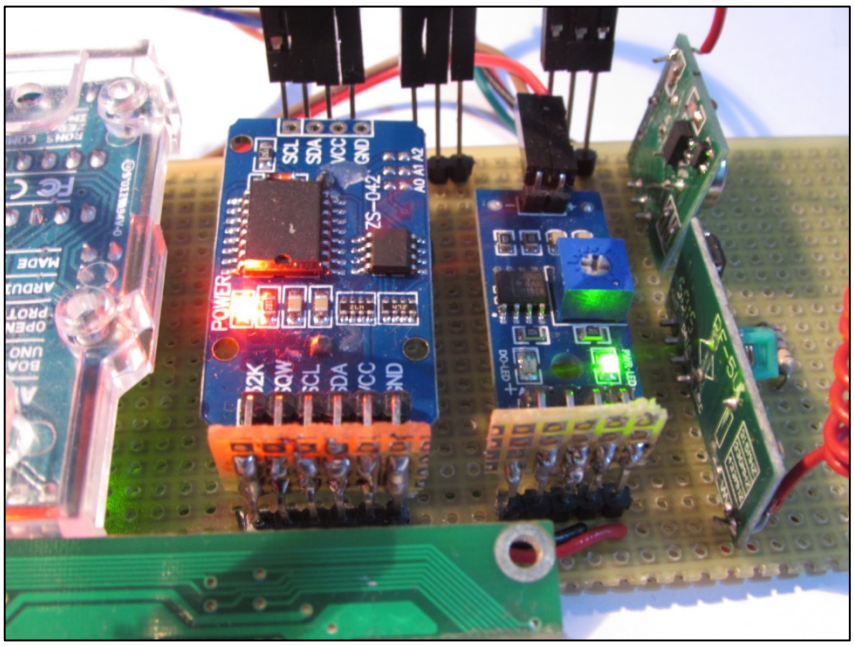


Figura 52 – Mòdul RTC i control pluja

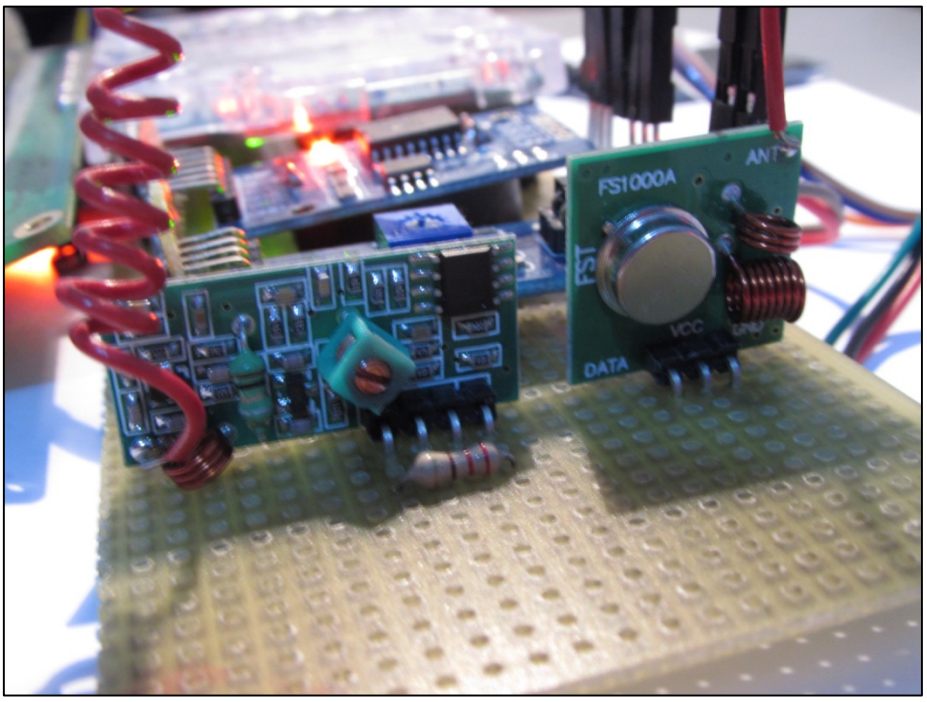


Figura 53 - Mòduls RF