Universitat Oberta de Catalunya

DISSENY I DESENVOLUPAMENT D'UNA ESTACIÓ METEOROLÒGICA AÏLLADA

Mateu Torrelló Martorell

Grau Tecnologies de la Telecomunicació

Febrer de 2016

DEDICATÒRIA I AGRAÏMENTS

Aquest projecte està dedicat a tota la comunitat que realitza l'esforç de compartir, recollir i crear informació de la plataforma Arduino.

Als creadors d'Arduino David Cuartielles i Massimo Banzi per desenvolupar una eina amb tant de potencial en l'àmbit de l'Art, Educació i Noves tecnologies.

Al Consultor Pere Tuset de la Universitat Oberta de Catalunya per haver-me permès portar a terme el desenvolupament del projecte sobre Arduino.

I sobretot agraïments a la meva família, parella i amics, els quals han suportat l'esforç que suposa estudiar un grau universitari a distància, a més de compaginar-ho amb la feina i vida personal.

FITXA DEL TREBALL FINAL

Títol del treball:	*Estació Meteorològica aïllada.*
Nom de l'autor:	*Mateu Torrelló Martorell*
Nom del consultor:	*Pere Tuset Peiró*
Data de lliurament:	*02/2016*
Àrea del Treball Final:	*Arduino*
Titulació:	*Grau Tecnologies de la Telecomunicació*

Resum del Treball:

El projecte consisteix en muntar i configurar una estació meteorològica aïllada, es crearà mitjançant Open Hardware basat amb la plataforma Arduino, que serà capaç de mesurar els principals canvis meteorològics i d'autoabastir-se amb energia solar. Juntament amb la placa Arduino s'implementarà un sistema de comunicació mòbil GSM per a poder enviar i rebre les dades mitjançant SMS.

Començarem per estudiar la situació actual de la cobertura mòbil en el territori nacional, i les diferents plaques Arduino que existeixen en el mercat junt amb els mòduls d'expansió anomenats "shields" compatibles amb aquesta plataforma. També especificarem els diferents sensors que seran necessaris per a captar els canvis meteorològics com son la temperatura, humitat relativa, pressió atmosfèrica, altitud, lluminositat i pluja. Finalment descriurem els components necessaris per a poder alimentar l'estació de manera autònoma amb una placa solar i una bateria Li-Po.

A continuació s'especificarà el procés de muntatge i programació de la placa Arduino perquè sigui capaç de realitzar les dues funcions principals:

La primera consisteix en ser capaç de llegir els SMS rebuts, processar-los i a continuació contestar el destinatari amb les dades pertinents.

La segona és la de prendre mesures del medi contínuament, per així poder activar una alarma en cas de que s'aveïnin pluges o tempestes.

El finalitzar el projecte obtindrem amb un sistema de control meteorològic aïllat i totalment automatitzat, d'aquesta manera ens permetrà accedir en temps real a les dades meteorològiques i també poder prevenir i alertar possibles canvis en el medi.

Abstract (in English):

The project consists in building and configuring an isolated weather station, created by the Open Hardware-based Arduino platform, which is able to measure the main meteorological changes and self-sufficiency with solar energy. With the Arduino board, a GSM mobile communication system will be implemented to send and receive data by SMS.

First of all, we begin by studying the current situation of mobile coverage in the country, and other Arduino boards that exist in the market along with the expansion modules called "shields" which is compatible with this platform. We will also specify the different sensors that will be needed to capture the meteorological changes such as temperature, relative humidity, atmospheric pressure, altitude and luminosity. Next, we will describe the components needed to feed the autonomously station with a solar panel and a Li-Po battery.

Then, we will specify the building process and programming the Arduino board to be able to perform two main functions:

Firstly, be able to read the SMS received, process them and then answer the addressee with appropriate data.

Secondly, take measures of environmental continuously, in order to activate an alarm in case of looming rains or storms.

Finally, at the end of the project, we will get an isolated and automated meteorological monitoring system that will allow us to access real-time weather data and then predict and alert possible changes in the environment.

Paraules clau:

Arduino, meteorològica, temperatura, humitat relativa, pressió atmosfèrica, GSM i SMS.

ÍNDEX

ÍNDEX DE IL·LUSTRACIONS

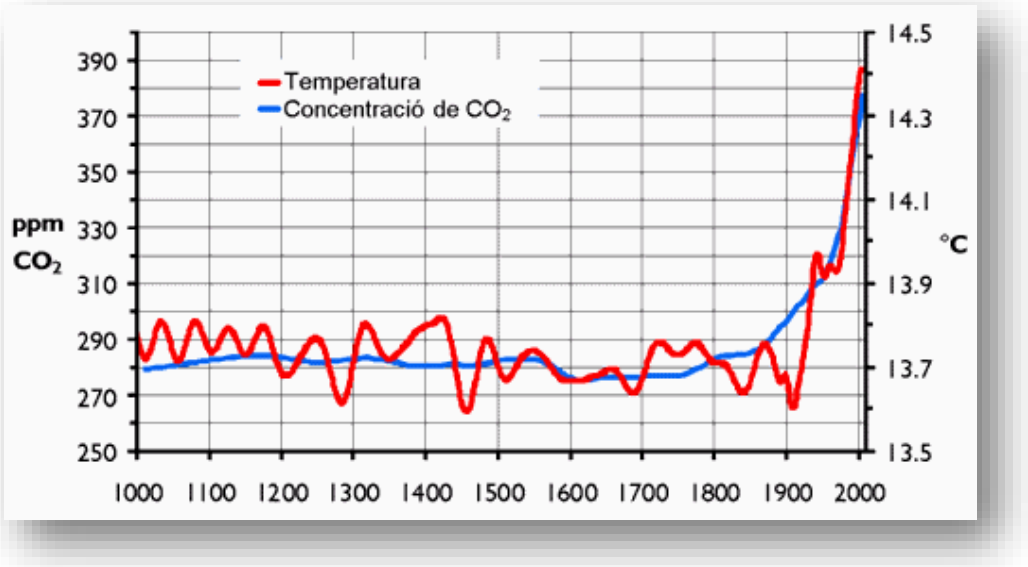
ÍNDEX DE TAULES

1. INTRODUCCIÓ

1.1 CONTEXT I JUSTIFICACIÓ DEL TREBALL

En l'actualitat, innombrables estudis científics han certificat de forma taxativa l'enorme influència que els fenòmens atmosfèrics tenen sobre la salut humana. No és només que una onada de fred provoqui episodis d'hipotèrmia o de congelació; o que una onada de calor ocasioni morts per cops de calor i deshidratació.
El que els estudis duts a terme posen de manifest és l'augment de morbiditat i de mortalitat que es deriva d'aquests fenòmens com a conseqüència, en gran nombre de casos, d'agreujament de malalties cròniques en persones vulnerables, sobretot gent gran.

El indubtable canvi climàtic al qual assistim, accelerat en les últimes dècades per l'augment exponencial de gasos d'efecte hivernacle a l'atmosfera, tindrà al llarg del segle acabat començat un impacte colossal sobre la salut de les persones, tant de forma directa, a l'augmentar en freqüència i durada dels fenòmens extrems, com indirectament en afectar l'hàbitat i les condicions socioeconòmiques de milions d'éssers humans.



Il·lustració 1: *Gràfic històric de temperatura i contaminació*

1.2 MOTIVACIÓ

Aquest treball sorgeix arran de la necessitat de crear una estació meteorològica que sigui capaç de transmetre'ns les dades obtingudes per el sensors, sense dependre de ninguna línia física de comunicacions i font d'energia externa.

Així doncs obrim un gran ventall de possibilitats a nivell científic, ja que podrem mesurar les dades meteorològiques en llocs remots on d'altra manera no seria possible.

1.3 OBJECTIUS DEL TREBALL

L'objectiu d'aquest treball és resumeix en dissenyar una estació meteorològica aïllada, capaç de mesurar la temperatura, humitat relativa, pressió atmosfèrica, altitud, lluminositat i detectar la pluja.

També s'implementarà una funció que sigui capaç de treballar de manera desatesa (standalone), i que respongui davant d'un SMS amb el text "Consulta" per enviar els valors meteorològics en temps real en qualsevol moment per el mateix medi de comunicació.

Per finalitzar estarà compost d'un petit sistema d'alarma sonora i visual que s'activarà en cas d'obtenir mesures límits i a continuació enviarà un SMS el destinaria pertinent.

- Familiarització amb la plataforma Arduino.
- Integració del grup de sensors d'àmbit meteorològic.
- Avaluació de la shield GSM/GPRS i les seves possibilitats de comunicació.
- Interconnexió del sistema global. Muntatge de la plataforma.
- Verificar i validar el correcte funcionament de l'estació i totes les seves funcionalitats.

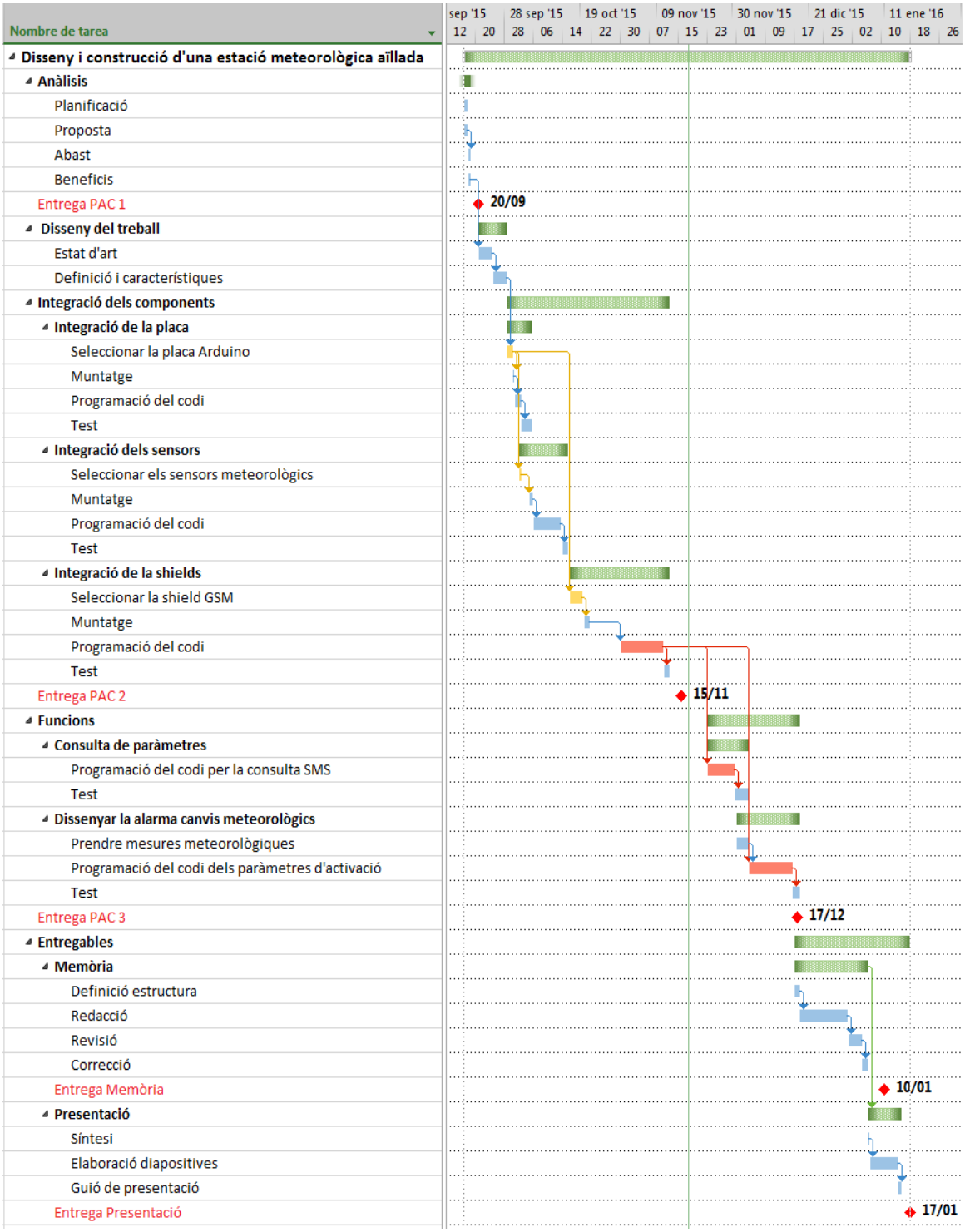
1.4 ENFOCAMENT I MÈTODE SEGUIT

Es tracte de un producte que existeix en el mercat, com son les estacions meteorològiques, però equipada amb un sistema de comunicació de consulta i alertes mitjançant SMS.
La metodologia per realitzar el Treball Final de Grau proporciona les mínimes garanties per assolir el desenvolupament del mateix en el termini establert i amb la màxima qualitat.
El desenvolupament s'ha realitzat utilitzant les metodologies àgil, utilitzant els següents conceptes:

- Desenvolupament iteratiu i incremental: desenvolupament basat en petites millores consecutives.
- Codi simple: implementació del codi de forma simple i anar afegint noves funcionalitats un cop verificades les anteriors.
- Proves continues: proves per verificar el correcte funcionament després de cada petit desenvolupament i la usabilitat de l'estació meteorològica.

1.5 PLANIFICACIÓ DEL TREBALL

En el següent diagrama de Gantt és pot observar la planificació del present Treball Final de Grau dividida en les principals tasques a realitzar:



Taula 1: *Diagrama de Gantt de la planificació del TFG*

1.6 BREU SUMARI DE PRODUCTES OBTINGUTS

- Arduino IDE principal API software.
- Shield SIMCOM GSM/GPRS API software.
- Sensor DHT22 API software.
- Sensor BMP085 API software.
- Sensor CMOS EL0405 API software.

1.7 BREU DESCRIPCIÓ DELS ALTRES CAPÍTOLS DE LA MEMÒRIA

L'estructura del present document és:

- **Capítol 2. Estat de l'art:** es detallen les característiques tècniques dels diferents sensors meteorològics, plataformes de computació i les shields GSM compatibles.
- **Capítol 3. Desenvolupament:** s'especifica la instal·lació i els paràmetres de configuració de tots els components que formen l'estació meteorològica.
- **Capítol 4. Desenrotllament:** es detallen les diferents funcionalitats que ofereix l'estació meteorològica.
- **Capítol 5. Conclusions:** es presenten les conclusions del present TFM i les possibles línies futures d'actuació que es deriven del projecte.
- **Capítol 6. Glossari:** Definició dels termes i acrònims més rellevants.
- **Capítol 7. Bibliografia:** Llista numerada de les referències bibliogràfiques utilitzades.

2. ESTAT DE L'ART

En aquest capítol es realitzarà un resum sobre l'estat de l'art de les diferents tecnologies implicades en el desenvolupament del projecte, les quals podrem associar amb els sensors meteorològics, les plataformes de computació i la connectivitat amb "Internet de les coses" juntament amb els sistemes de comunicacions GSM, GPRS i la utilització d'ordres Hayes (AT).

2.1 SENSORS METEOROLÒGICS

Els primers tipus d'estacions meteorològiques automàtiques, molts d'ells d'exploració sensible electromecànica, estaven equipats amb sensors mecànics: termòmetre bimetàl·lic, higròmetre de cabell, baròmetre de càpsula aneroide, sensor de precipitació de catúfol basculant, etc.

A les estacions automàtiques s'utilitzen els següents sensors, classificats per ordre de prioritat:

Paràmetre mesurat	Tipus de sensor
Temperatura	Termòmetre de resistència de platí
	Termistor, resposta lineal
	Termistor, resposta exponencial
	Termòmetre bimetàl·lic
Humitat	Cèl·lula de rosada de licitació
	Psicròmetre de termistor
	Higròmetre de cabell
	Higròmetre Pernix
Pressió atmosfèrica	Càpsula aneroide
	Baròmetre de mercuri
Velocitat del vent	Sensor de roda de cassoleta
	Sensor d'hèlix
Direcció del vent	Penell
Precipitació	Catúfol basculant
	Cambra volumètrica de vàlvula
	Balança de pesar
Heliofanía	Sensor de fotocèl·lula
	Sensor de termistor

Radiació solar	Termopila
Detector de luminància	Fotocèl·lula
Detector de precipitació	Sensor de resistència elèctrica
	Sensor de capacitància elèctrica
Sostre de núvols	Ceilòmetre de feix giratori
Visibilitat	Transmisòmetro
	Visibilímetre de retrodispersió

Taula 2: *Tipus de sensors meteorològics*

La fiabilitat del sensor per usar amb un equip automàtic pot decidir-se considerant dos jocs de característiques del sensor:

- Les característiques funcionals del sensor, és a dir exactitud, estabilitat de les característiques de calibratge, especificitat de resposta i durabilitat.

- Els trets d'acoblament a l'equip automàtic del sensor: sortida eventual o analògica, algoritme del paràmetre atmosfèric al sensor, fàcil intercanviabilitat, requisits de manteniment mòdics, requisits d'energia econòmics.

A continuació explicarem els instruments més comuns que s'utilitzen en les estacions meteorològiques professionals, les seves unitats, rang i precisió mínimes de mesura.

- **Termòmetre:** Instrument que mesurar la temperatura de l'aire, la escala de mesura més utilitzada és del graus centígrads (ºC), on 0 ºC correspon el punt de congelació de l'aigua, i 100 ºC el punt de ebullició de l'aigua. També existeixen altres variants com a termòmetres de sota terra i de temperatura arran de terra.

  Rang i precisió mínimes d'un termòmetre: -20 ºC a 60 ºC $\pm$ 1 ºC.

- **Psicròmetre:** Mesura la humitat relativa de l'aire. Aquest instrument conté un termòmetre de bulb humit i un termòmetre de bulb sec, i la humitat relativa de l'aire és la diferencia de temperatura entre els dos termòmetres mesurat així en tant per cent (%). Quant la humitat relativa és del 100% significa que el aire esta format per vapor d'aigua.

  Rang i precisió mínimes d'un psicròmetre: 10% a 90 % $\pm$ 5 %.

- **Baròmetre:** Mesura la pressió atmosfèrica en la superfície. La pressió atmosfèrica és el pes per unitat de superfície exercida per la atmosfera. El mercuri és uns dels baròmetres més utilitzats. La unitat utilitzada en aquest instrument és el hectopascal (hPa), hecto vol dir cent i pascal unitat de mesura de pressió. La pressió atmosfèrica mitja en la terra és de 1013.25 hPa.

  Rang i precisió mínimes d'un baròmetre: 500 hPa a 1050 hPa $\pm$ 0.1 hPa.

- **Altímetre:** Mesura l'altitud respecte el nivell del mar. La unitat utilitzada és el metre (m).

  Rang i precisió mínimes d'un altímetre: -300 m a 5000 m $\pm$ 1 m.

- **Heliògraf:** Mesura el temps d'il·luminació solar durant un dia, mesurat amb hores (h). Existeixen de dos tipus: per rotació de la terra i per rotació mecànica.

  Rang i precisió mínimes d'un heliògraf: 0 h a 24 h $\pm$ 0.04 h.

- **Pluviòmetre:** Mesura la quantitat d'aigua que cau en un metre quadrat de superfície. La unitat de mesura és el litre per metre quadra (l/m^2).

  Rang i precisió mínimes d'un pluviòmetre: 0 l/m^2 a 15 l/m^2 $\pm$ 0.2 l/m^2.

- **Anemòmetre:** Mesura la velocitat del vent amb un petit moli de tres aspes, la unitat utilitzada és kilòmetres per hora (km/h).

Rang i precisió mínimes d'un anemòmetre: 0 km/h a 150 km/h ± 1 km/h.

- **Penell:** Mesura la direcció del vent amb un dispositiu giratori situat sobre una placa que indica els punt cardinals, mesurat en graus (º).

  Rang i precisió mínimes d'un penell: 0 º a 360 º ± 4.5 º.

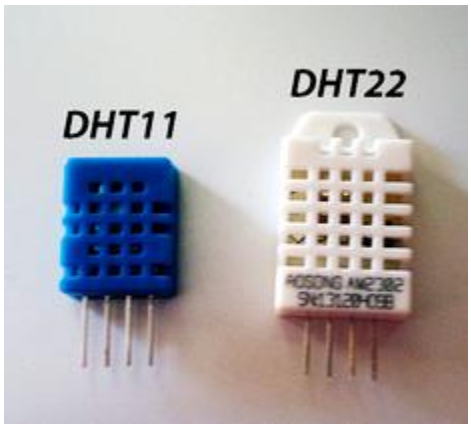
- **Piranòmetre:** Mesura la radiació solar que incideix sobre la superfície de la terra. Es mesura en kilowatts per metre quadrat (KW/m²) en un camps de 180 graus.
  Existeixen dos tipus de piranòmetre: tèrmic i fotovoltaic, format per una pila termoelèctrica, i per un fotodíode respectivament.

  Rang i precisió mínimes d'un piranòmetre: 0 KW/m² a 3000 KW/m² ± 0.02 KW/m².

Així doncs els sensors meteorològics que utilitzarem en el nostre dispositiu seran el següents: temperatura (Termòmetre), humitat relativa (Psicròmetre), pressió atmosfèrica (Psicròmetre), altitud (Altímetre), llum (Heliògraf) i un sensor de pluja.

2.1.1 SENSOR DE TERMÒMETRE I PSICRÒMETRE

Després de llegir i comparar les característiques tècniques dels principals sensors de temperatura i humitat, em vaig decidir per aquesta gama de sensors DHT (DHT11 i DHT22) que contenen els sensors de temperatura i humitat units en un mateix dispositiu, oferint molt bones prestacions.



Il·lustració 2: *Sensor DHT11 i DHT22*

Físicament, posseeixen algunes diferències, entre les quals es destaca la diferència de color i la mesura de l'encapsulat.

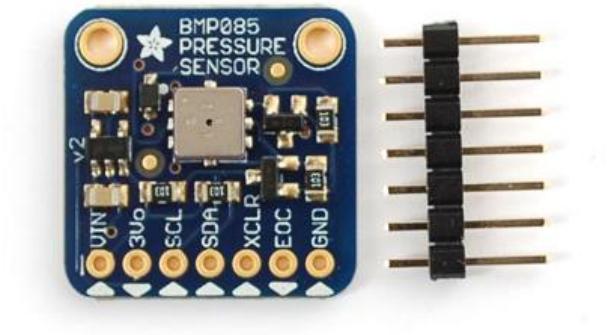
Model	DHT11	DHT22
Tensió de operació	3-5 VDC 2.5 mA max	3-5 VDC 2.5 mA max
Rang de temperatura	0 a 50 ºC	-40 a 80 ºC
Precisió de temperatura	± 2 ºC	± 0.5 ºC
Rang de humitat	20-90 % HR	0-100 % HR
Precisió de humitat	±5 % HR	±2 % HR
Mesures decimals	No	Si

Taula 3: *Comparativa sensor DHT11 i DHT22*

En la tabla anterior es pot observar que el DHT22 és molt més precís que el DHT11, la qual cosa em fa decidir clarament fer ús d'aquest sensor per el nostre dispositiu. Datasheet[1].

2.1.2 SENSOR DE PSICRÒMETRE I ALTÍMETRE

En quant el sensor de pressió atmosfèrica i altitud me decidit per el model BMP085 de la marca Bosch perquè ofereix gran precisió i prestacions.
A causa de que la pressió canvia amb l'altitud també es pot utilitzar com a altímetre i calcular la altitud respecte el nivell del mar. El sensor està soldat a un PCB i a un regulador de 3.3V, un canviador de nivell I2C i resistències de pull-up en els pins I2C. Datasheet[2].



Il·lustració 3: *Sensor BMP085*

Característiques:

- Vin: 3.3 VCC.
- I2C 7 bits de adreça 0x77.
- Rang de mesurament de pressió: 300 a 1100 hPa.
- Precisió de pressió: 0.03 hPa.
- Rang de mesurament de altitud: -500 a 9000 m a nivell del mar.
- Precisió de altitud: 0.25 m.
- Rang de temperatura de treball: -40 ºC a 85 ºC.

2.1.3 SENSOR DE PLUJA

En un principi no sabia que existien sensors de pluja, el vaig conèixer per casualitat cercant models dels sensors anterior, però quant el vaig trobar me va parèixer molt interessant per poder activar una alerta de pluja a l'estació meteorològica.

He escollit el model CMOS EL0405. Esta formada per dues parts, una placa resistiva que detecta les gotes de pluja a l'entrar en contacte amb la placa i produeix una certa circulació de corrent quan sigui excitat i esta fabricada amb materials especialment antioxidants. L'altre part es situa el controlador EL0405 acompanyat d'un potenciòmetre que ens permet ajustar la sensibilitat de la senya que s'envia a l'Arduino. Datasheet [3].

Il·lustració 4: *Sensor EL0405*

Característiques:

- Tensió de treball: 3,3V-5V
- Sortida analògica i digital
- Mesura de la placa base: 3.2x1.4 cm
- Mesura de la placa sensor: 5x4 cm
- Indicador a la placa per mitjà de LED de presència/absència de pluja.

2.1.4 SENSOR DE HELIÒGRAF

Per simular aquest instrument, detectar la intensitat de la llum, amb utilitzar un fotodíode. Un fotodíode és un semiconductor construït amb una unió PN, sensible a la incidència de la llum visible o infraroja. Perquè el seu funcionament sigui correcte es polaritza inversament, de manera que es produirà una certa circulació de corrent quan sigui excitat per la llum. A causa de la seva construcció, els fotodíodes es comporten com cèl·lules fotovoltaiques, és a dir, il·luminats en absència una font exterior d'energia generen un corrent molt petita amb el positiu a l'ànode i el negatiu al càtode.

Vin: 5 VCC
Corrent: 10 mA a 100 mA
Diàmetre: Ø8mm

Il·lustració 5: *Fotodíode*

I finalment a l'hora d'activar la alarma sonora i visual he decidit utilitzar un brunzidor Buzzer passiu i un díode LED verd.

Vin: 3 a 12 VCC
Impedància: 16 Ω
Freqüència de treball: 2 a 5 KHz

Il·lustració 6: *Buzzer*

Vin: 3.4 a 4 VCC
Corrent: 20 a 30 mA
Diàmetre: Ø5mm

Il·lustració 7: *díode LED*

2.2 PLATAFORMES DE COMPUTACIÓ

Les mini controladores o mini PC's són sempre una bona opció per gaudir de tota la potència d'un ordinador però recorrent a una mesura compacte. Els podem utilitzar com a servidor de continguts, connectats al televisor i, per descomptat, com a ordinador a l'ús.

2.2.1 PLATAFORMA RASPBERRY PI

Raspberry Pi és un dels productes més populars per aquests fins, tant pel seu atractiu preu com per les enormes opcions que porta amb si.
El 2012 va cridar l'atenció de desenes de milers d'entusiastes, després emergir com un ordinador de baix cost que pogués arribar al major nombre d'usuaris possible, i gràcies a l'àmplia comunitat que aporta valor en aquest projecte, hi ha usos tan diversos com senzills de implementar amb uns pocs coneixements.

Raspberry Pi, es tracta d'una diminuta placa base de 85 x 54 mil·límetres (una mica més gran que un paquet de tabac) en el qual s'allotja un xip Broadcom BCM2835 amb processador ARM fins a 1 GHz de velocitat, GPU VideoCore i fins 512 Mbytes de memòria RAM. Quant al seu preu, sol estar per sota dels 40 euros, una de les raons que explica la seva popularitat.

Perquè funcioni, només cal que afegim nosaltres mateixos un mitjà d'emmagatzematge, com ara una targeta de memòria SD, endollar al corrent gràcies a qualsevol carregador de tipus microUSB.
La fundació de Raspberry Pi[4] posa a disposició des de la seva pàgina web Raspbian, una distribució de Linux basada en Debian, però també podem recórrer a moltes de les distribucions específiques que la comunitat d'usuaris ha desenvolupat per a diversos fins.

En funció del model que escollim, disposarem de més o menys opcions de connexió, però sempre tindrem almenys un port de sortida de vídeo HDMI i un altre de tipus RCA, minijack d'àudio i un port USB 2.0 al qual connectar un teclat i ratolí.

Pel que fa a connexió de xarxa, podem disposar d'Ethernet per endollar un cable RJ-45 directament al Router.

Il·lustració 8: *Raspberry Pi*

Actualment hi ha dos models de Raspberry Pi. El més popular és el Raspberry Pi Model B+, que ve amb processador ARM 1176JZF-S a 700 MHz, dos ports USB 2.0 i un d'Ethernet, sent la resta de les seves característiques les mateixes que us hem avançat al principi d'aquest apartat.

Per altre banda, el model Raspberry Pi 2 Model B que és la més actual, incrementant la velocitat del processador fins a 900 MHz i duplica la capacitat de la memòria RAM fins a 1 GB, tot i així mantenint el mateix preu.

	Model B+	2 Model B
SoC	Broadcom BCM2835	Broadcom BCM2836
CPU	700MHz ARM1176JZF-S	900MHz Quad-core ARM Cortex-A7
GPU	VideoCore IV	VideoCore IV
RAM	512Mb	1Gb
USB	4	4
Vídeo	Jack, HDMI	Jack, HDMI
Audio	Jack, HDMI	Jack, HDMI
Boot	MicroSD	MicroSD
Red	Ethernet 10/100	Ethernet 10/100
Consum	500mA / 2,5w / 5v	800mA / 4w / 5v
Alimentació	MicroUSB / GPIO	MicroUSB / GPIO
Tamany	85 x 56 mm	85 x 56 mm
Preu	35$	35$

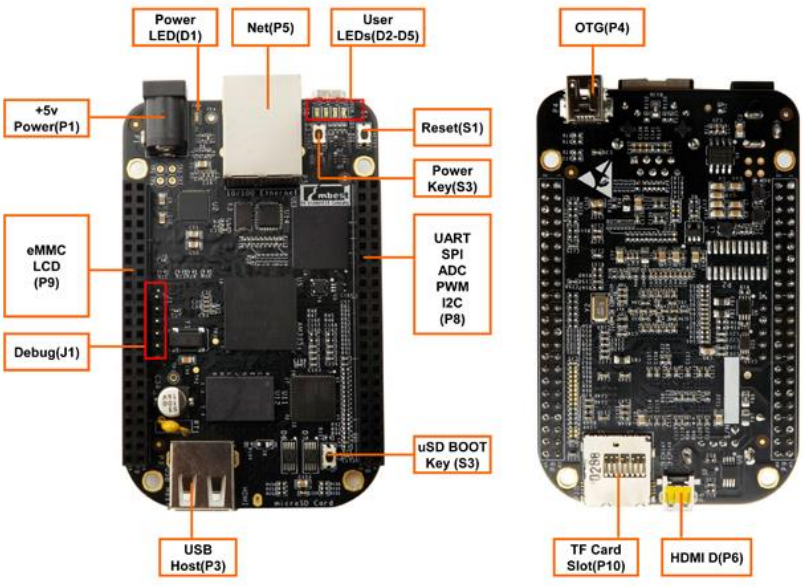
Taula 4: *Models Raspberry Pi*

2.2.2 PLATAFORMA BEAGLEBONE

BeagleBone és un ordinador petit de la mesura d'una targeta de crèdit, on pots executar un sistema operatiu, com pot ser Linux / Android 4.0.
Igual que Raspberry Pi seva principal diferència amb Arduino és que pot executar un petit sistema operatiu, és pràcticament un miniordinador on es poden executar programes.

BeagleBone, està dissenyat per funcionar a un nivell molt més alt i té molta més capacitat de procés que Arduino. La placa va ser desenvolupada per un petit equip d'enginyers com una placa educacional que podria ser usada en col·legis al voltant del món per ensenyar les capacitats del programari i maquinari lliure. Des de la seva aparició ha anat evolucionant mitjançant models nous que incorporen certes millores en rendiment i connectivitat. Avui en dia el model més recent és el BeagleBone Black, el llançament va ser a l'abril de 2013.

Un altre dels avantatges més destacables de la BeagleBone Black és la connectivitat que ofereix. Disposa d'un total de 92 pins reconfigurables que inclouen fins a quatre ports sèrie diferents, la generació de 8 PWM independents, 4 temporitzadors, 1 bus CAN, 7 entrades analògiques de 1.8 V amb una resolució del convertidor AD de 12 bits, diverses sortides d'alimentació a 5 i 3.3 V, dues connexions SPI, dos ports I2C per a la seva interconnexió amb qualsevol tipus de maquinari compatible, a més de 65 entrades i sortides digitals reconfigurables. Les extensions per a les targetes Beagleboard s'anomenen Capes i hi ha més de 40 models diferents.



Il·lustració 9: *BeagleBone Black*

2.2.3 PLATAFORMA ARDUINO

A continuació entrem en detall sobre els diferents elements que conformaran el sistema basat en la plataforma Arduino. Començarem amb una breu presentació del que és en si aquesta plataforma, i analitzarem els diferents models de plaques disponibles al mercat. També s'avaluaran les diferents shields (mòduls d'expansió) GSM/GPRS compatibles amb aquesta plataforma, i els principals sensors de temperatura que tenim al nostre abast.

Què és Arduino? Arduino és una plataforma electrònica de maquinari lliure basada en una placa amb un microcontrolador. Amb programari i maquinari flexibles i fàcils d'utilitzar, Arduino ha estat dissenyat per adaptar-se a les necessitats de tot tipus de públic, des d'aficionats, fins a experts en robòtica o equips electrònics. També consta d'un simple, però complet, entorn de desenvolupament, que ens permet interactuar amb la plataforma de manera molt senzilla. Es pot definir per tant com una senzilla eina de contribució a la creació de prototips, entorns, o objectes interactius destinats a projectes multidisciplinaris i multitecnologia.[5] A la següent il·lustració podem veure les seves plaques més venudes a nivell mundial, es tracta del model Arduino UNO.

A través d'Arduino podem recollir multitud d'informació de l'entorn sense excessiva complexitat. Gràcies als seus pins d'entrada, ens permet jugar amb tota una gamma de sensors (temperatura, lluminositat, pressió, etc.) que ens brinden la capacitat de controlar o actuar sobre certs factors de l'entorn que l'envolten, com ara: controlant llums , accionant motors, activant alarmes... i molts altres actuadors.

Gràcies al fet que la plataforma és de maquinari lliure, les plaques Arduino poden ser fetes a mà per qualsevol aficionat o comprades ja muntades de fàbrica.

Pel que fa al programari, és totalment gratuït, i està disponible per a la descàrrega per qualsevol interessat en la pròpia pàgina web de Arduino [6]. L'entorn de desenvolupament disposa d'un propi llenguatge de programació per al microcontrolador de la placa Arduino, basat en Processing / Wiring.

Una dels principals motius pel qual resulta molt interessant la utilització de la plataforma Arduino per a determinats projectes, es basa en la seva independència respecte a haver de

mantenir-se connectat a un PC. Arduino és perfectament capaç de treballar en mode "standalone", només cal assegurar-nos d'haver carregat prèviament el programa que volem que mantingui en execució. Si bé, tot això no el priva de poder operar mantenint en tot moment la connexió amb el PC, sent capaç de comunicar-se amb diferents tipus de programari, com ara: Macromedia Flash, Processing, Max / MSP, Pure Data, etc.

Des del moment de la seva creació, allà per l'any 2005, quan Arduino va néixer com un projecte educatiu, les innovacions no han deixat de succeir-se. [7] A dia d'avui hi ha multitud de plaques Arduino, i la majoria d'elles estan disponibles en diferents versions, adaptables pràcticament a qualsevol tipus de requisits o necessitats per dur a terme un determinat projecte. Els principals models de plaques Arduino que podem trobar al mercat a dia d'avui són els següents:

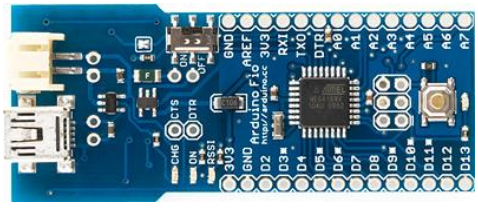
UNO: És l'última versió de la placa bàsica d'Arduino. Es connecta a l'ordinador mitjançant un cable USB estàndard i conté tot el necessari per començar a programar i utilitzar la placa. Les seves funcionalitats es poden veure incrementades gràcies al fet que hi ha multitud de shields perfectament compatibles amb aquest model. A diferència de l'antiga Duemilanove, integra un nou xip USB-sèrie i compta amb un nou disseny d'etiquetatge, facilitant la identificació de les diferents entrades i sortides de la placa. Amb un preu aproximat de 24 €, es tracta pot ser de la placa Arduino més interessant a l'hora de buscar la millor relació preu-prestacions, de manera que serà un dels models a tenir molt en compte per al desenvolupament del projecte.

Il·lustració 10: *Arduino UNO*

MEGA: L'Arduino Mega és probablement la placa amb majors prestacions de la família Arduino. Compta amb 54 pins digitals, que funcionen com a entrada / sortida, a més de les seves 16 entrades analògiques. És la placa més gran i potent d'Arduino. És totalment compatible amb les shields Arduino UNO, i compta amb una memòria que duplica la seva capacitat en comparació amb la resta de plaques. Arduino MEGA és per tant l'opció més adequada per a aquells projectes en què es requereixi un gran nombre d'entrades i sortides disponibles, o per suportar la càrrega de codis de programació pesats que no poden emmagatzemar-se en les memòries de menor capacitat que ofereixen altres plaques , com per exemple el model UNO. El seu preu és de 45 €.

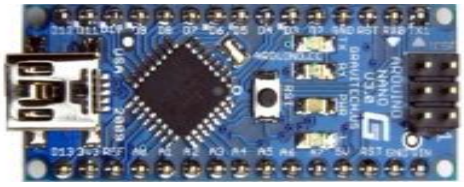
Il·lustració 11: *Arduino MEGA*

Fito: Un Arduino orientat per al seu ús a manera de node sense fils. Posseeix connectors per a la integració d'un mòdul XBee, i disposa d'un connector per a bateria de liti i un circuit per carregar la bateria. Preu 23 €.



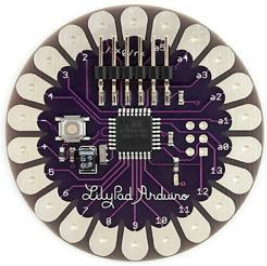
Il·lustració 12: *Arduino Fito*

Nano: Una placa compacta dissenyada per utilitzar directament en plaques de desenvolupament. La seva connexió amb el PC ha de ser a través d'un cable Mini-B USB. Preu: 40 €.



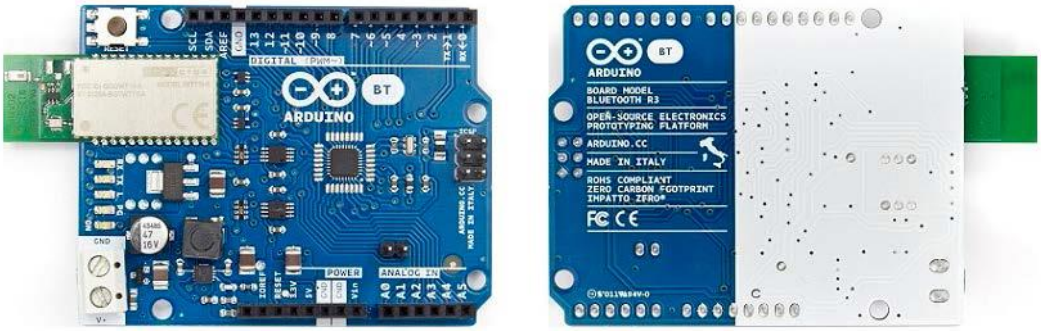
Il·lustració 13: *Arduino Nano*

LilyPad: Una model de la placa circular, de mesura reduïda, compacta i dissenyada específicament per a ser cosida a la roba o qualsevol tipus de material flexible, orientat a qualsevol tipus d'aplicacions. Necessita d'un adaptador addicional per a la seva comunicació amb el PC. Preu: 20 €.



Il·lustració 14: *Arduino LilyPad*

Bluetooth: L'Arduino BT conté un mòdul Bluetooth que permet comunicar-se i programar sense necessitat de cablejat.

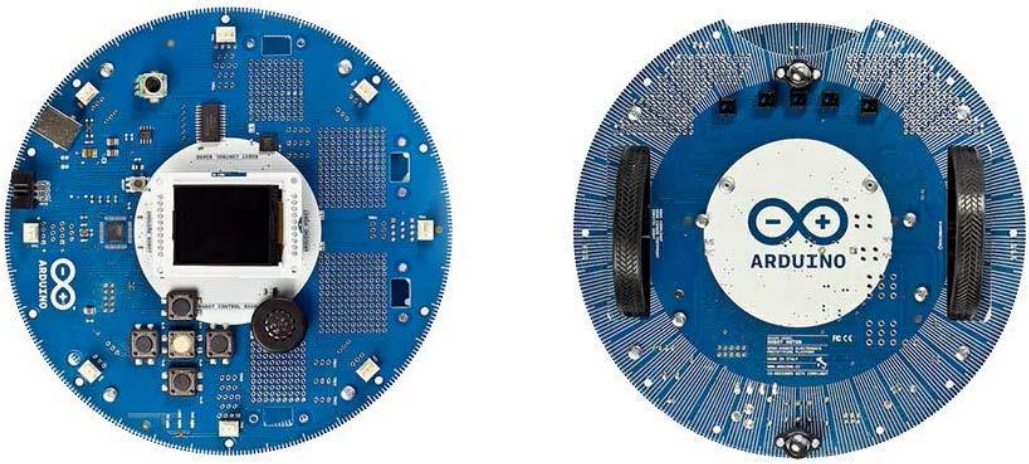


Il·lustració 15: *Arduino Bluetooth*

Ethernet: Simplement es tracta d'una placa Arduino, de l'estil a Arduino UNO, però dotada amb un port Ethernet per a la seva connexió a Internet. És una solució interessant per a aquells projectes en què es necessiti estar connectat permanentment a Internet. Preu: 35 €.

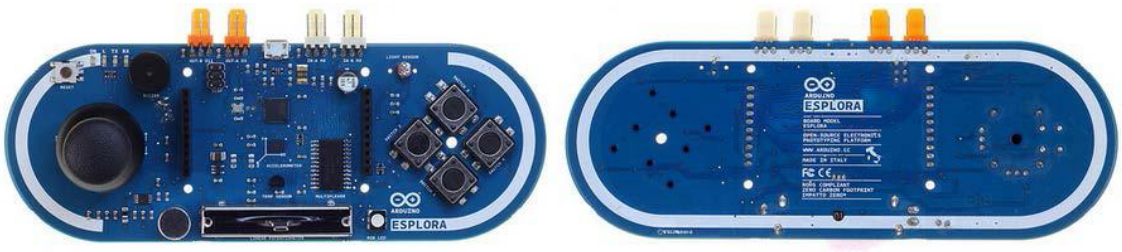
Il·lustració 16: *Arduino Ethernet*

Robot: Aquest innovador model és la primera placa oficial Arduino sobre rodes. El robot consta de dos processadors, un per cadascuna de les seves dues plaques. Té una placa dedicada al control dels motors, i l'altra és l'encarregada de processar les dades rebudes per part dels sensors i decidir com ha d'actuar en tot moment. El model Robot té molts dels seus pins ja assignats als sensors i actuadors de bord, i la seva programació és molt similar al procés requerit pel Arduino UNO. Queda clar que es tracta d'una placa destinada a la robòtica. Preu: 200 €.



Il·lustració 17: *Arduino Robot*

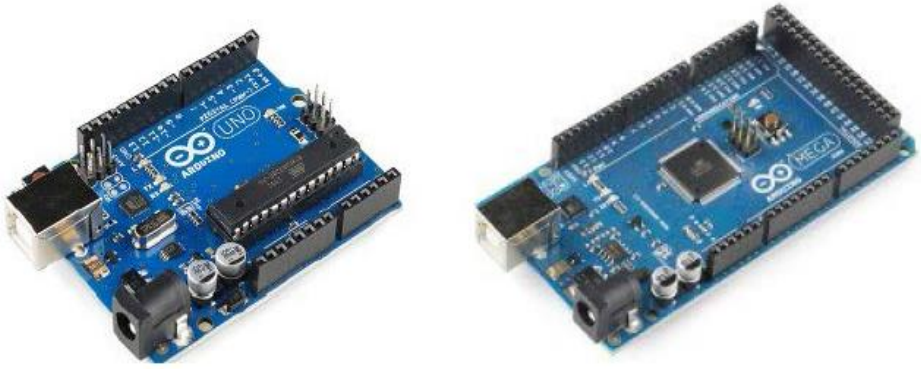
Explora: És una placa derivada del model Leonardo, diferent a totes les altres plaques Arduino precedents. A punt per utilitzar sensors multitud de sensors a bord, ha estat dissenyada per a aquelles persones que vulguin començar a jugar amb Arduino sense haver de posseir certs coneixements mínims en electrònica. Explora incorpora multitud de funcionalitats: un dispositiu capaç d'emetre sons, un micròfon, sensors de llum, sensors de temperatura, fins i tot un acceleròmetre i una palanca a manera de comandament de control. A més, permet estendre les seves capacitats gràcies a les seves dues entrades TINKERKIT, connectors de sortida, i port per a la connexió de pantalla TFT LCD. Podem trobar-la per uns 55 €.



Il·lustració 18: *Arduino Explora*

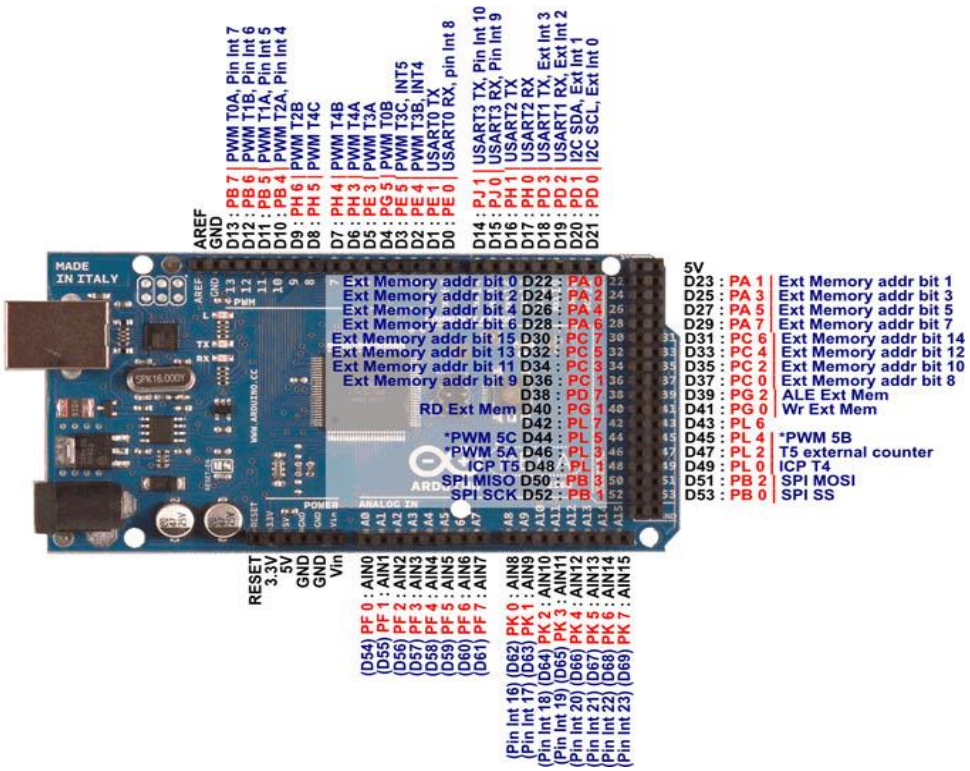
Finalment triarem la placa adequada per crear l'estació meteorològica. El preu i mesura de tots els dispositius son pràcticament iguals, la Raspberry Pi és 40 vegades més ràpid que un Arduino quant es tracte de velocitat de rellotge però això només quant es tracte de aplicacions de software, en el nostre cas elegirem la placa Arduino ja que fa que aquesta sigui molt millor en projectes de hardware el compta en més entrades analògiques i digitals, t la possibilitat de controlar múltiples dispositius.

Inicialment vaig estudiar la possibilitat de realitzar el sistema amb l'Arduino UNO, però després vaig decidir que per la poca diferencia de cost, valia més la pena utilitzar l'Arduino MEGA. En primer lloc, la memòria de programa de l'Arduino UNO és de 32 KB, bastant limitada si volem desenvolupar una aplicació que ja dugui integrades varies funcionalitats. Arduino MEGA en canvi, disposa de 256 KB de memòria de programa, per el qual obtenim un marge bastant major a l'hora de generar un sketch més complet.
Una de les altres raons principals per elegir l'Arduino MEGA és que conté 50 pins més respecte a l'Arduino UNO (10 analògics i 40 digitals), facilitant així les connexions dels diferents elements i la possibilitat d'escalar les prestacions del projecte en un futur.



UNO		MEGA	
Microcontroller	ATmega328	Microcontroller	ATmega2560
Operating Voltage	5V	Operating Voltage	5V
Input Voltage (recommended)	7-12V	Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V	Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)	Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	6	Analog Input Pins	16
DC Current per I/O Pin	40 mA	DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA	DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader	Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	2 KB (ATmega328)	SRAM	8 KB
EEPROM	1 KB (ATmega328)	EEPROM	4 KB
Clock Speed	16 MHz	Clock Speed	16 MHz

Taula 5: *Comparativa placa Arduino UNO i MEGA*

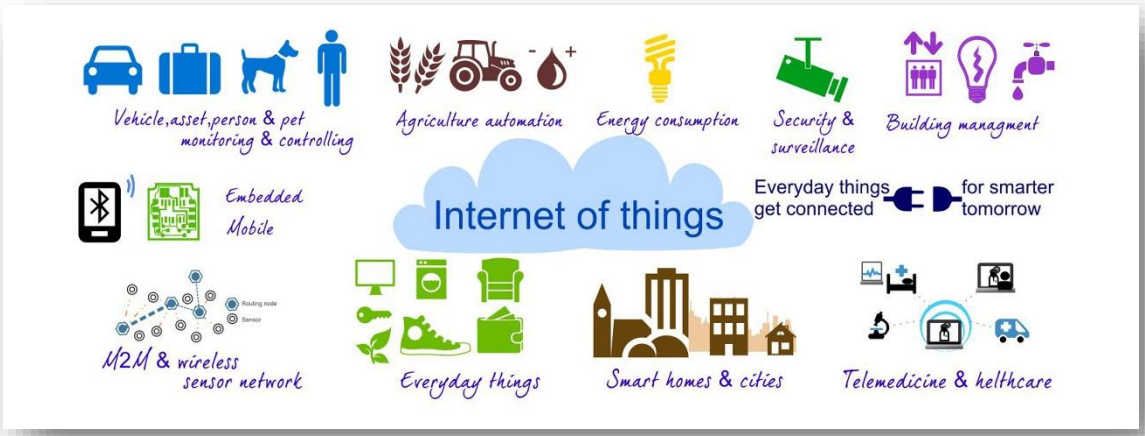


Il·lustració 19: *Esquema de connexions Input/Output Arduino MEGA*

2.3 SISTEMES DE COMUNICACIÓ

2.3.1 INTERNET OF THINGS (IOT)

L'expressió "Internet of Things" fa referència a la interconnexió entre objectes de consum o d'ús quotidià (electrodomèstics, roba, llibres, productes Alimentaris, etc.) a través de certs dispositius capaços de connectar-los a la xarxa.

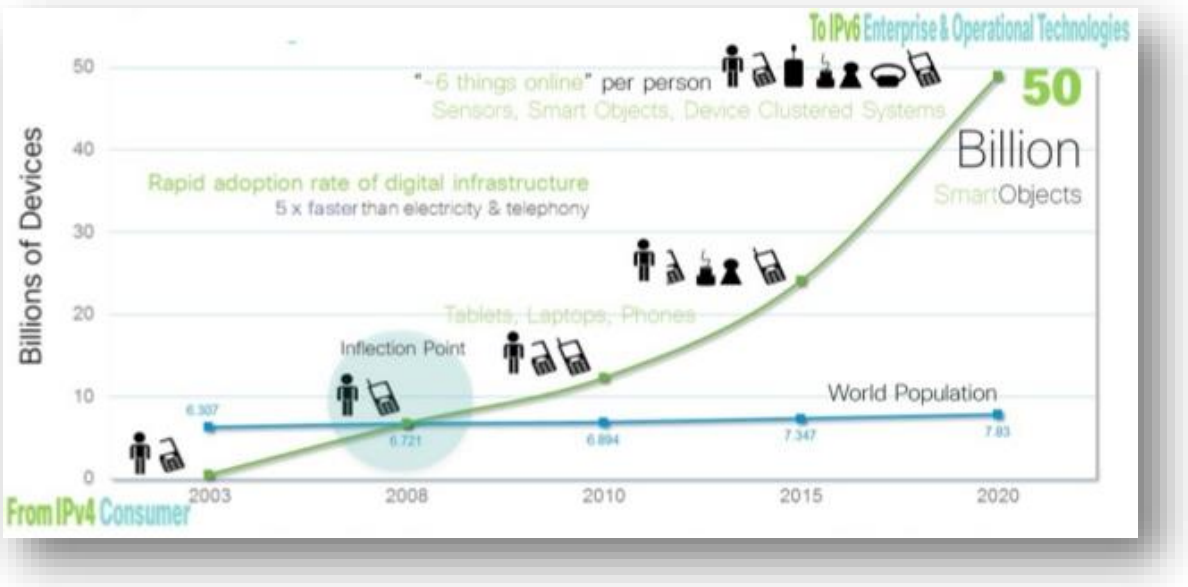


Il·lustració 20: *Varietat de contingut Internet of Things*

Les utilitats de Internet de les coses poden considerar-se pràcticament infinites. Cada vegada són més els mitjans que tenim a la nostra disposició per implementar, de manera no molt complexa, un possible dispositiu capaç de dotar cert objecte o funció d'una connexió a la xarxa, ja sigui per mitjà d'una tecnologia o una altra. L'arribada d'IPv6 suposa un factor clau en el desenvolupament del concepte IoT. Gràcies el nou protocol (dissenyat per reemplaçar a IPv4) s'evitarà que el creixement de Internet quedi restringit, i farà possible la gestió d'adreçament d'innombrables dispositius.

D'altra banda, ja són moltes les aplicacions mòbils i serveis en el núvol que ens permeten la connexió a tots aquests dispositius, i proporcionen una via per al tractament d'una immensa quantitat de dades en temps real (sistemes Big Data), facilitant la integració d'infinitat de sensors aplicables pràcticament a qualsevol tipus de necessitat. De fet, ja han sorgit fins i tot xarxes socials de sensors, com la plataforma "Xively"[8], on els usuaris

comparteixen dades en temps real procedents de diferents sensors. En els propers anys s'espera un gran augment en el nombre d'equips d'ús quotidià interconnectats, entre altres coses, gràcies a la imminent arribada de tots aquests sensors intel·ligents a les nostres llars[9].



Il·lustració 21: *Projecció de creixement de Internet of Things fins a l'any 2020*

També són moltes les empreses que ofereixen, o estan interessades, en solucions IoT. La gestió de recursos i l'eficiència energètica són les aplicacions més demanes. Tecnologies sense fils com les xarxes mòbils, WIFI, Zigbee, Bluetooth, etc., permeten optimitzar possibles solucions i faciliten el seu desplegament.

Sense anar més lluny, Arduino ha marcat un punt d'inflexió en aquest sentit, convertint-se en l'eina ideal per dur a terme multitud de prototips i obtenir nous caos d'ús. Gràcies al seu baix cost, senzillesa i la varietat de models que podem trobar, resulta una eina de gran ajuda a l'hora d'implementar idees i solucions d'àmbit domèstic. A més, existeixen multitud de sensors i actuadors compatibles amb aquesta plataforma, mitjançant els quals podem recollir dades del nostre entorn, analitzar-los, i actuar en conseqüència, fins i tot connectar amb altres dispositius a través de les diferents tecnologies de comunicació (GSM / GPRS, 3G, Bluetooth, RFID, etc.) aprofitant tota una varietat de shields d'expansió.

2.3.2 SISTEMA GSM

GSM és l'abreviatura de “Sistema Global per a les comunicacions Mòbils” (en anglès, Global System for Mobile communications). Al començament del segle XXI, és l'estàndard més utilitzat d'Europa [10]. Conegut com a estàndard de segona generació (2G), la seva principal diferència respecte a la primera generació de telèfons mòbils és que les seves comunicacions són totalment digitals.

L'estàndard GSM va ser desenvolupat a partir de 1982, quan va ser estandarditzat per primera vegada, denominat "Groupe Spécial Mobile". Va sorgir com a idea per al desenvolupament d'un estàndard europeu de telefonia mòbil digital. El 1991 es va convertir en un estàndard internacional anomenat "Sistema Global de Comunicacions Mòbils", i van començar a presentar-se els primers prototips de telefonia GSM.

A Europa, el sistema GSM utilitza les bandes de freqüència de 850, 900 i 1800 MHz, mentre que als Estats Units és fa servir la banda de freqüència de 1900 MHz. En conseqüència, els dispositius de comunicacions mòbils que poden operar tant a Europa com als Estats Units es coneixen com quatribanda (Quadband).

L'estàndard GSM permet transmissions digitals de veu i dades, com missatges de text (SMS) o missatges multimèdia (MMS). Respecte a la seva arquitectura de xarxa, en GSM tot terminal mòbil ha d'estar constituït per una targeta SIM (mòdul d'identificació d'abonat) i el mateix dispositiu, normalment un telèfon mòbil [11].

La targeta SIM és l'encarregada d'identificar a la xarxa a l'usuari i al terminal mòbil. Aquests dispositius s'identifiquen gràcies a un nombre exclusiu d'identificació denominat IMEI (Identificador internacional d'equips mòbils), compost per 15 dígits.

D'altra banda, cada targeta SIM també posseeix un número d'identificació únic anomenat IMSI (Identificador internacional d'abonats mòbils). A la següent il·lustració podem veure l'arquitectura de xarxa corresponent al sistema GSM. Està composta per múltiples estacions base (BTS), que al seu torn, es connecten a un controlador d'estacions base (BSC), encarregat de l'administració de la xarxa.

Aquest sistema compost pel BSC i els seus corresponents estacions base connectades el mateix, se li coneix com BSS (Subsistema d'estacions base).
En un nivell superior estarien els Centres de commutació mòbil (MSC), al qual es connecten físicament els controladors d'estacions base. Aquest és l'encarregat d'establir la connexió amb la xarxa de telefonia pública i amb Internet.

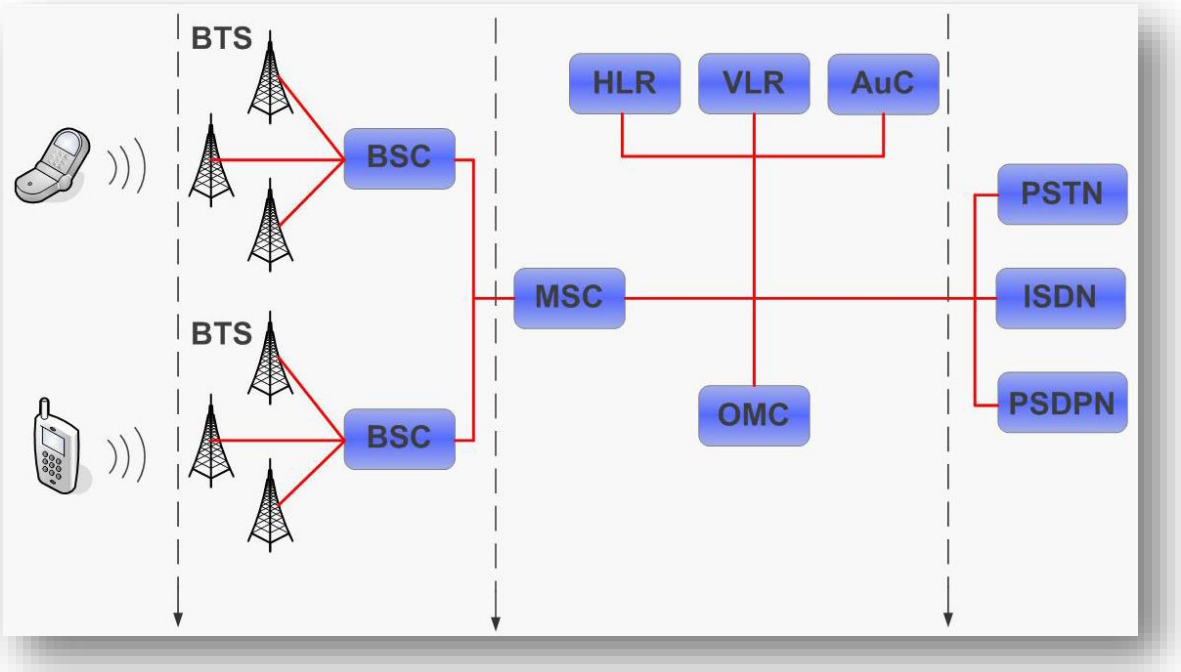
La seva administració va a càrrec de l'operador de la xarxa telefònica. El MSC pertany a un Subsistema de commutació de xarxa (NSS), el qual s'encarrega d'identificar els usuaris, determinar la seva ubicació, i gestionar les comunicacions amb altres usuaris de la xarxa. En aquest moment el Centre de Commutació mòbil (MSC) es connecta a una sèrie de base de dades que li proporcionen funcions addicionals:

- Registre de posició base (HLR): en aquesta base de dades s'emmagatzema la informació dels abonats (posició geogràfica, informació administrativa, etc.).

- Registre de posició visitant (VLR): conté informació d'usuaris que no són abonats locals. Les dades es conserven mentre l'usuari està dins de la zona, i s'eliminen quant l'abandonen o després d'un llarg període d'inactivitat.

- Registre d'identificació de l'equip (EIR): és una base de dades que conté la llista amb els dispositius mòbils.

- El Centre d'autentificació (AUC): la seva missió és verificar les identitats dels usuaris.



Il·lustració 22: *Arquitectura de xarxa corresponent al sistema GSM*

Actualment podem observar a la següent il·lustració que el nivell de cobertura és el més elevat de totes les comunicacions mòbils, ja que avarca el 98% del territori nacional [12]:

Il·lustració 23: *Cobertura GSM (2G) en Espanya*

2.3.3 SISTEMA GPRS

L'estàndard GPRS o Servei General de Paquets via Ràdio (en anglès, General Packet Radio Service) és una evolució del sistema GSM. És també conegut com GSM ++, però atès que es tracta d'un estàndard de telefonia mòbil intermedi entre la segona generació (2G) i la tercera (3G), sovint rep la nomenclatura de 2.5G [13].

GPRS estén l'arquitectura de l'estàndard GSM per permetre la transferència de dades mitjançant commutació de paquets amb velocitats de transferència que ronden els 114 Kbps.

Al contrari del que passa en commutació de circuits, en l'estàndard GPRS, gràcies a la seva manera de transferència de paquets, les transmissions de dades només utilitzen la xarxa quan cal, permetent la tarifació per volum d'informació transmesa en lloc de per temps de connexió, per tant, l'usuari pot estar connectat sense cost addicional, ja que només utilitzarà la xarxa quan enviï o rebi un paquet de dades.
Per a l'accés a la xarxa de dades, l'estàndard GPRS utilitza el protocol IP, mentre que per al transport de veu, empra l'arquitectura de la xarxa GSM.

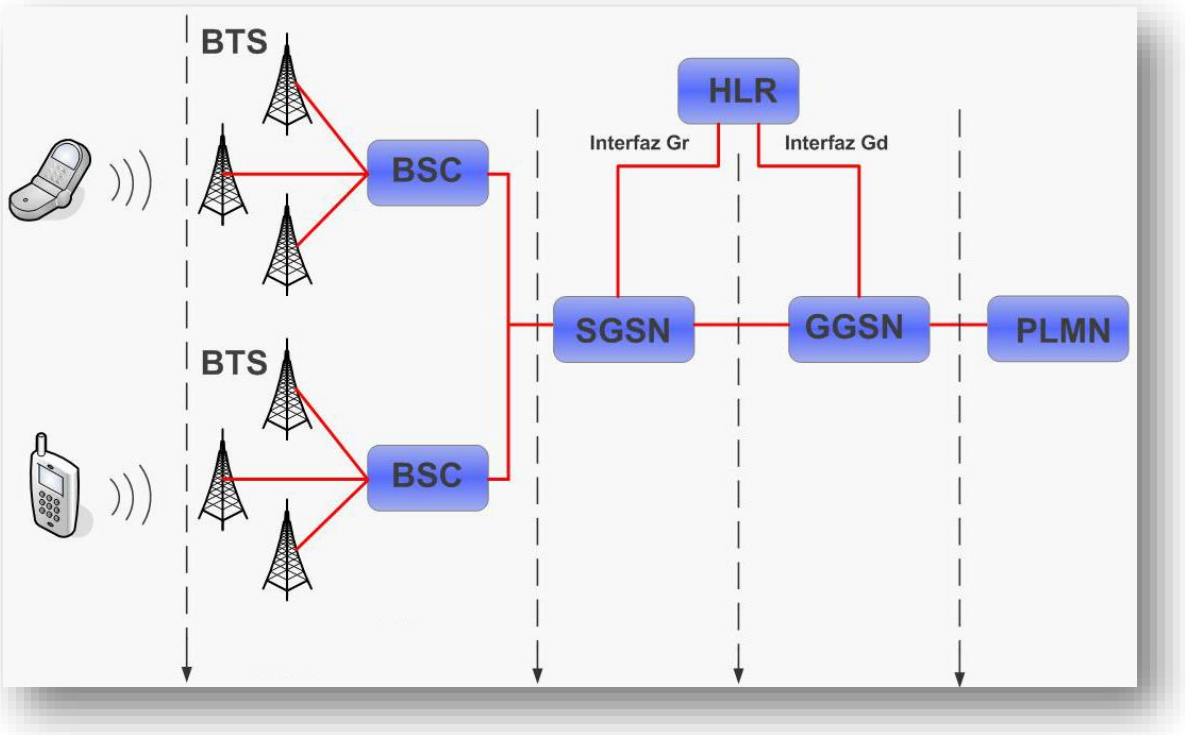
A part d'actualitzar alguns serveis amb els que ja comptava GSM, la tecnologia GPRS admet una altra sèrie de característiques que no estaven disponibles en 2G:

- Serveis de missatges curts (SMS) - Serveis de missatges multimèdia (MMS)
- Servei punt a punt (PTP); per a la connexió client
- Servidor en una xarxa IP
- Servei punt a multipunt (PTMP); per a l'enviament de multidifusió.

A la pròxima il·lustració es mostra l'estructura funcional del sistema GPRS, basada en l'addició de nous nodes sobre la infraestructura corresponent a GSM. Aquests nodes se'ls coneix com GSN (nodes de suport GPRS):

- El SGSN (Node de suport de serveis GPRS) s'encarrega de l'encaminament dels paquets IP entrants i sortints dirigits cap, o procedents, de qualsevol abonat GPRS situat dins de la zona geogràfica a la qual dóna servei aquest SGSN.

- EL GGSN (Node de suport passarel·la GPRS) serveix d'interfície cap a les xarxes externes de paquets IP. Encamina les adreces IP dels abonats servits per la xarxa GPRS, intercanviant informació d'encaminament amb la xarxa externa. El GGSN prepara la comunicació amb xarxes externes i gestiona les sessions de GPRS. També inclou funcions per associar abonats amb la SGSN que correspongui. També recull dades de tarifació i ús de la xarxa de dades externa.



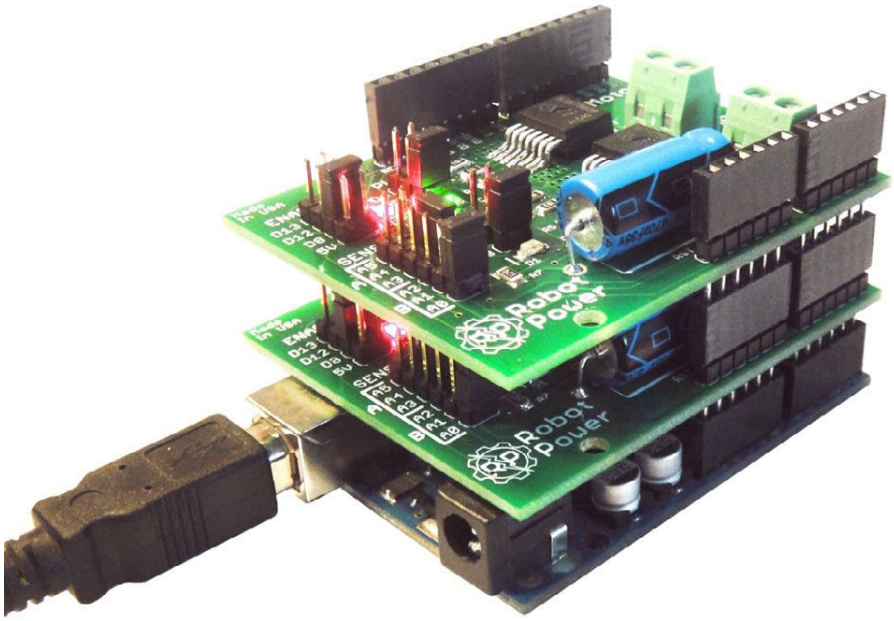
Il·lustració 24: *Arquitectura de xarxa corresponent al sistema GPRS*

Actualment podem observar a la següent il·lustració que el nivell de cobertura és bastant elevat, ja que avarca el 82% del territori nacional [12]:

Il·lustració 25: *Cobertura GPRS (3G) en Espanya*

2.3.4 SHIELDS GSM

Si volem ampliar les funcionalitats de la nostra plataforma Arduino, sempre podem recórrer a una gran varietat de shields compatibles pràcticament amb qualsevol dels seus models. D'aquesta manera, podem dotar el dispositiu de funcions addicionals dedicades específicament a oferir algun tipus de servei concret. Un shield és un mòdul d'expansió en forma de placa impresa que es pot connectar a la part superior de la placa Arduino per ampliar les seves capacitats, permetent a més de ser apilades unes sobre d'altres mantenint un disseny modular, tal com podem veure a la següent il·lustració.



Il·lustració 26: *Disseny modular format per un Arduino i dues shields*

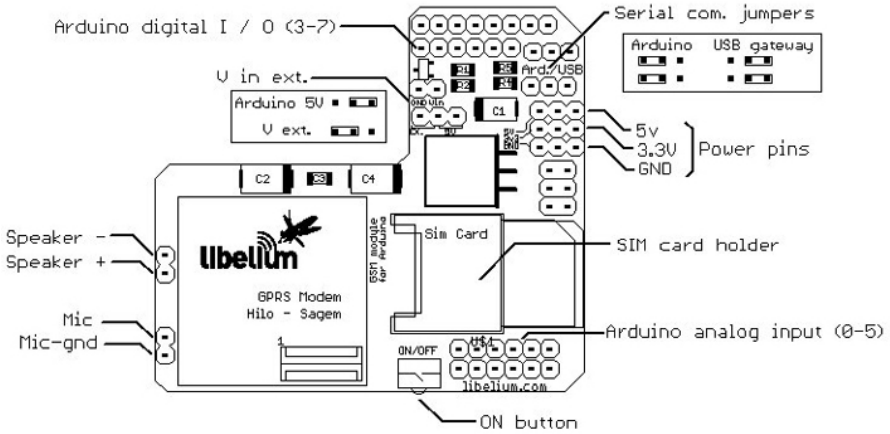
En el nostre cas, necessitem una shield que ens permet utilitzar els sistemes de comunicacions mòbils per poder interactuar a distància amb la nostra plataforma. Navegant per Internet podem trobar diverses shields que han estat dissenyades específicament per oferir serveis a través dels sistemes GSM, GPRS, 3G, o una combinació d'aquests. A més, són perfectament compatibles amb la nostra placa Arduino[14].

Mòdul GPRS Quadband per Arduino:

Aquesta placa integra un mòdul FIL SAGEM que ens permet oferir els serveis d'un mòdem GPRS a través del nostre Arduino. Amb aquesta shield, podem enviar SMS o fer trucades a altres dispositius mòbils. És necessària la connexió d'una antena externa per poder establir les comunicacions. El preu d'aquesta shield és aproximadament d'uns 86€.

Il·lustració 27: *Mòdul GPRS Quadband per Arduino*

En l'esquema reflectit a la pròxima il·lustració apareix indicada la funció de cada un dels ports que componen la placa:



Il·lustració 28: *Diagrama de connexions del mòdul GPRS Quadband*

En principi es tracta d'un mòdul que s'adapta perfectament al que estàvem buscant, però més endavant veurem que hi ha plaques més econòmiques que ofereixen exactament les mateixes funcionalitats, de manera que inicialment ha quedat descartada.

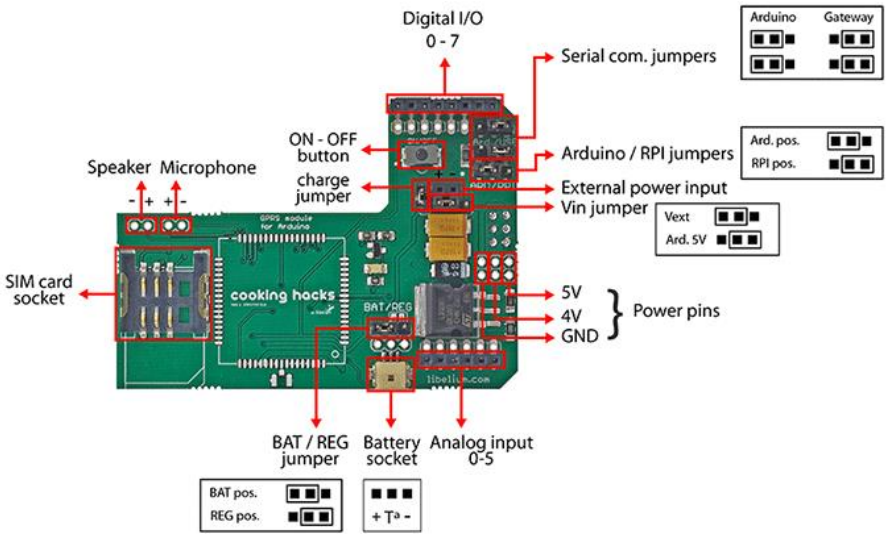
Mòdul GPRS + GPS Quadband per Arduino / Raspberry Pi

Aquest és l'últim model de la shield GPRS per Arduino. Gràcies al fet que compta amb un mòdul SIM908 integrat en la pròpia placa, ofereix la possibilitat d'utilitzar la tecnologia GPS per posicionament en temps real, resulta molt útil per a aquelles aplicacions en les que necessitem conèixer la ubicació del nostre dispositiu.

Il·lustració 29: *Mòdul GPRS + GPS Quadband per Arduino / Raspberry Pi*

Gràcies a aquest mòdul d'expansió compatible amb la majoria de les plaques Arduino, no només podem transmetre la ubicació del sistema a través d'HTTP i emmagatzemar-la en un servidor web, sinó que també podem utilitzar Google Maps per veure la seva localització sobre el mapa en tot moment.

Igual que la resta de plaques d'aquest estil, l'antena s'ha d'adquirir per separat, només que en aquest cas necessitaríem també una segona antena específica per GPS. El preu d'aquesta shield és aproximadament d'uns 120 €.



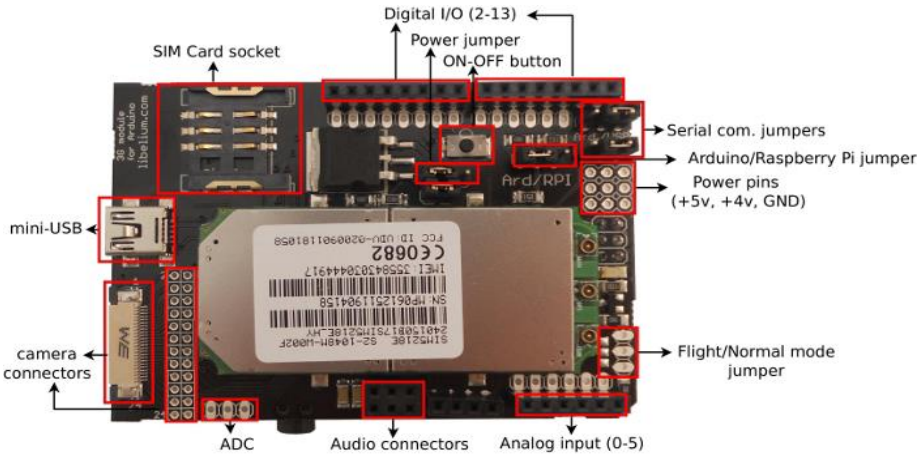
Il·lustració 30: *Diagrama de connexions del mòdul GPRS + GPS Quadband*

En el nostre cas, no tenim intenció d'utilitzar la tecnologia GPS, per tant no té sentit decantar-nos per aquesta placa. A més, amb el nostre plantejament de reduir els costos en la mesura de lo possible, no encaixaria molt escollir aquesta shield tenint altres models més econòmics que ofereixen estrictament els serveis que necessitem. Si bé, per a aquells projectes en què es necessita ubicar la posició del dispositiu, pot ser un model molt interessant, i podrem tenir-ho en compte a l'hora d'oferir funcions addicionals per al nostre sistema en desenvolupament.

Mòdul 3G + GPRS + GPS per Arduino / Raspberry Pi

Aquest és el model més complet entre totes les shields GPRS disponibles. A part del sistema GPRS, gràcies al seu mòdul SIM5218, integra també serveis 3G i tecnologia GPS. El seu preu és bastant elevat, aproximadament 180 €, però és cert que admet més funcionalitats en comparació amb la resta de shields que hem vist fins ara, fins i tot permet la connexió d'una càmera per a la presa d'imatges. A la pròxima il·lustració podem veure l'aspecte que presenta aquesta shield.

Il·lustració 31: *Mòdul 3G + GPRS + GPS per Arduino / Raspberry Pi*



Il·lustració 32: *Diagrama de connexions del mòdul 3G + GPRS + GPS*

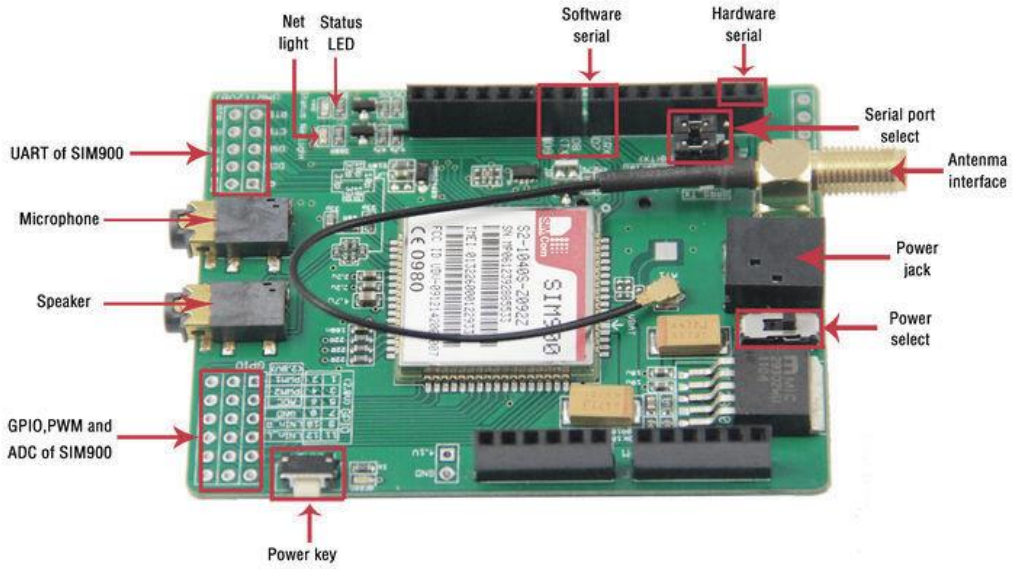
Tot i que és tracta de la placa amb majors prestacions d'entre totes les que hem pogut trobar, no te cap sentit decantar-nos per ella, doncs a part de tenir un preu molt elevat, inclou tecnologies de les quals no farem cap us (GPS i 3G). Desestimem per tant, l'opció d'adquirir aquesta shield.

Mòdul GSM/GPRS Quadband per Arduino

Aquesta mòdul de la marca SIMCOM pot convertir la nostra placa Arduino en un plataforma capaç d'oferir connectivitat GSM/GPRS. Integra un mòdul SIM900 que ens permet establir trucades amb altres dispositius mòbils, enviar SMS, fins i tot la comunicació de dades a través dels protocols TCP, UDP, HTTP o FTP. El seu preu és molt interessant és aproximadament d'uns 50 €.

Il·lustració 33: *Mòdul GSM/GPRS per Arduino*

No hi ha dubte que es tracta d'un dels mòduls més barats que podem trobar al mercat de les shields per a l'ús de sistemes de comunicacions mòbils. No només te un preu adequat, sinó que ens ofereix tots els serveis que necessitem per poder dur a terme el desenvolupament del nostre dispositiu. Finalment, ja que es tracta de la placa amb millor relació preu-prestacions que hem pogut trobar, ens vam decantar per aquest model per integrar-lo en la nostra estació meteorològica.



Il·lustració 34: *Diagrama de connexions del mòdul GSM/GPRS*

Característiques:

- Quad-Band 850/900/1800/1900 MHz.
- GPRS multi-slot classe 10/8.
- GPRS classe d'estació mòbil B.
- Control a través de ordres AT.
- Rellotge RTC.
- Port sèrie seleccionable.
- Connectors mini-jack de altaveu i auriculars.
- Baix consum d'energia 1.5mA (en mode repòs).
- Rang de temperatura de treball -40Cº a +85 Cº.

2.3.5 ORDRES AT

Les ordres AT, també coneguts com a ordres Hayes (en honor al seu desenvolupador Dennis Hayes), són una sèrie d'instruccions que conformen una interfície de comunicació entre usuari i mòdem. La seva abreviatura AT per la qual són mundialment coneguts aquestes ordres prové de la paraula “attention”.

Tot i que la finalitat principal de les ordres AT va ser la comunicació amb mòdems, la telefonia mòbil GSM/GPRS també va adoptar aquest llenguatge com a estàndard de comunicació. En l'actualitat, tots els terminals mòbils GSM tenen una sèrie específica de ordres AT que ens permeten configurar-los per mitjà d'aquestes instruccions i indicar un seguit d'accions que volem que s'executin, com ara marcar un número de telèfon, enviar o llegir un SMS, consultar l'estat de connexió a la xarxa, llegir o escriure a l'agenda de contactes, etc.
Gràcies al fet que la transmissió d’ordres AT no depèn del canal de comunicació a través del qual aquests siguin enviats (cable, infrarojos, Bluetooth, etc.), podrem utilitzar
la nostra placa Arduino per transmetre aquests ordres a un mòdul GSM/GPRS que sigui capaç d'interpretar i actuar en conseqüència.

Com que són molts els ordres existents, únicament anem a esmentar juntament amb una breu descripció, aquells que puguin resultar de principal interès en el desenvolupament del projecte :

- **AT:** Amb aquesta ordre verificarem que el mòdul està rebent les nostres instruccions. Si tot és correcte, ha de respondre amb un "OK". En cas contrari no obtindrem resposta.

- **AT + CPIN?:** Utilitzarem aquesta instrucció per comprovar si la nostra targeta SIM està desbloquejada o si per contra, requereix introduir el codi PIN per a procedir amb el desbloqueig de la mateixa. Quan la SIM estigui operativa respondrà amb un "ready".

- **AT + CPIN = "****":** En el cas que necessitem introduir el PIN, aquest és el ordre que hem d'enviar, escrivint els 4 dígits corresponents al codi de desbloqueig en el lloc dels asteriscs, delimitat entre cometes.

- **AT + CREG?:** Amb aquesta ordre preguntem per l'estat de connexió a la xarxa. La resposta rebuda seguirà la següent notació: + CREG: <n>, <stat>, on:

  <stat> = 0 → No registrat, no està buscant una connexió de xarxa
  <stat> = 1 → Registrat a la xarxa nacional
  <stat> = 2 → No registrat, però buscant la xarxa
  <stat> = 3 → Registre denegat
  <stat> = 5 → Registre tipus roaming

- **AT + COPS?:** Mitjançant aquesta instrucció rebrem la confirmació de la companyia en la qual està registrada la nostra targeta SIM (Movistar, Orange, Vodafone, etc.)

- **AT + CMGF = <f>:** Selecciona el format del missatge SMS, on:

  <f> = 0 → manera PDU
  <f> = 1 → manera text

- **AT + CSCS = "IRA":** Amb aquesta ordre seleccionem el set de caràcters que és utilitzat per l'enviament d'SMS. En el nostre cas ens interessa configurar el mòdul en mode "IRA" (International Reference Alphabet).

- **AT + CMGS = "*********":** A través d'aquesta ordre marcarem el número de mòbil al qual volem fer arribar l'SMS. Anirà delimitat per cometes.

- **AT + CMGR = <r>:** És l'ordre utilitzat per llegir SMS emmagatzemats a la memòria de la targeta SIM. En lloc de <r> posarem el número corresponent a la posició del missatge que volem llegir. Per exemple, si volem llegir l'últim missatge rebut, posarem una '1', si volem llegir el penúltim missatge posarem un '2', i així successivament.

- **AT + CPMS = "SM":** Amb aquesta instrucció seleccionem el directori del que volem llegir un missatge de text. En funció de les sigles que posem després del igual, podem recollir la informació de diferents memòries. En aquest cas, amb les sigles "SM", seleccionem com a directori la memòria de la targeta SIM.

- **AT + CMGD = <n>:** Aquesta ordre serveix per eliminar SMS de la memòria de la targeta SIM. Substituirem <n> per la posició en la memòria que ocupi el SMS que volem esborrar. Per exemple, per esborrar el missatge emmagatzemat en la memòria de la SIM la instrucció seria AT + CMGD = 1.

- **AT + CMGL = "ALL":** Amb aquesta ordre podem veure una llista de tots els SMS que tinguem a la SIM, tant llegits, com no llegits.

- **+ CMTI: "SM", <pos>:** Aquesta és la instrucció que rebrem automàticament per part del mòdul quan es rebi un SMS, on <pos> és el número corresponent a la posició en memòria en què s'ha emmagatzemat dit missatge . Per exemple, si tenim la memòria de la SIM buida i ens arriba un SMS, la instrucció que ens enviaria el mòdul seria + CMTI: "SM", 1.

Nota: Per a l'enviament de qualsevol dels ordres AT a través de Hyperterminal o l'entorn de desenvolupament d'Arduino, hem de seleccionar sempre els caràcters fi de línia i retorn de carro.

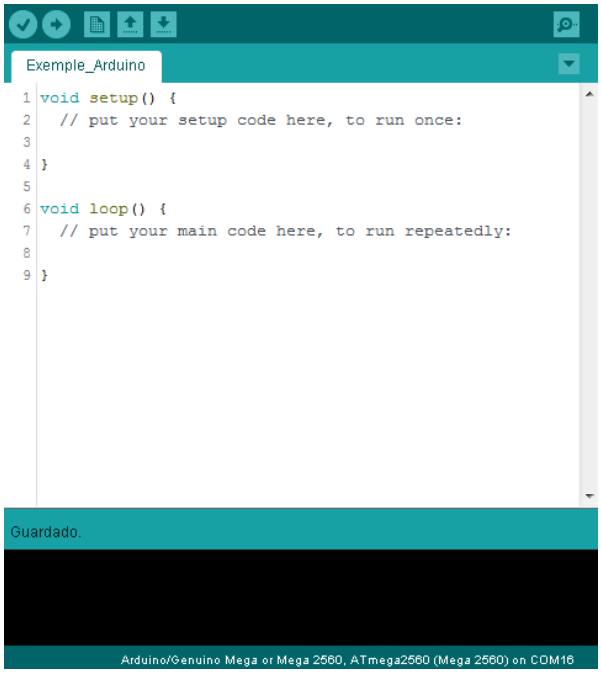
3. DESENVOLUPAMENT

En aquest apartat s'exposen pas a pas cadascuna de les etapes per les que s'han anat passant fins arribar a l'objectiu final del projecte, inclou: proves realitzades, codis de programació corresponents a cadascuna de les funcions que s'han anat implementant.

3.1 ENTORN DE DESENVOLUPAMENT A ARDUINO

L'entorn de desenvolupament en Arduino (IDE) és l'encarregat de la gestió de la connexió entre el PC i el maquinari d'Arduino per tal d'establir una comunicació entre ells per mitjà de la càrrega de programes. Com podem veure a la proxima il·lustració, l'IDE d'Arduino es compon de [15]:

- Un editor de text: on escriure el codi del programa.
- Una àrea de mensajes: a través del qual l'usuari tindrà constància en tot moment dels processos que es trobin en execució, errors en el codi, problemes de comunicació, etc.
- Una consola de text: mitjançant la qual podrem parlar amb la maquinari Arduino i viceversa.
- Una barra d'eines: on podrem accedir a una sèrie de menús i als botons amb accés directe a les principals funcionalitats d'Arduino.



Il·lustració 35: *Arduino IDE*

A través de la IDE d'Arduino, podem escriure el codi del programa i crear el que es coneix com "sketch" (programa). Per què en diem sketch i no programa? Doncs perquè la IDE d'Arduino ve de Processing, i en aquest llenguatge de programació enfocat al món gràfic, cada codi és considerat un esbós, en anglès "sketch".

El sketch permet la comunicació amb la placa Arduino. Aquests programes són escrits a l'editor de text, el qual admet les possibilitats de tallar, enganxar, cercar i reemplaçar text.

A l'àrea de missatges es mostra, tant la informació mentre es carreguen els programes, com els possibles errors que tinguem a l'hora de compilar, ja sigui per problemes en el codi del sketch, per error en la detecció del nostre Arduino al port USB, o per qualsevol altre problema que sigui detectat.

La consola mostra el text de sortida per l'entorn d'Arduino incloent els missatges d'error complets i altres informacions.

Des de la barra d'eines tenim accés directe a les principals funcionalitats que ofereix la IDE d'Arduino, com per exemple: verificar el procés de càrrega, crear un nou sketch, obrir un sketch ja existent, guardar els programes, obrir el Monitor Serial, etc.

Dins dels menús, cal esmentar l'existència de llibreries, que poden proporcionar funcionalitats addicionals per a la utilització en sketches, per exemple per treballar amb maquinari o manipular dades. Per utilitzar una llibreria dins d'un sketch, hem de declarar-la prèviament. Per a això ens anirem al menú "sketch", i seleccionarem l'opció importar llibreries. Dins buscarem la llibreria que sigui del nostre interès i la importarem al sketch, inserint una sentència de tipus “#include” al començament del mateix. S'ha de tenir en compte que en carregar un codi que inclogui llibreries, aquestes també es bolquen en la placa juntament amb la resta de l'esquetx, incrementant l'ocupació del programa i reduint l'espai disponible a la memòria de l'Arduino.

A continuació passem a descriure la utilitat de cada una de les icones que apareixen a la pantalla principal de l'entorn de desenvolupament de l'Arduino:

Verificar: Aquesta funcionalitat s'encarrega de verificar el codi del sketch i la recerca de possibles errors. A través de l'àrea de missatges se li notificarà a l'usuari el resultat de la verificació. En el cas que es detectin errors en el codi, aquests es detallaran juntament amb el número de línia en què han estat detectats. Només quan la comprovació resulta lliure d'errors podrem procedir a la càrrega del codi en la nostra placa Arduino.

Carregar: Permet compilar el codi del sketch i el carrega en Arduino. Quan la càrrega a acabat s'informa a l'usuari a través de l'àrea de missatges, i podrem procedir a l'obertura del monitor serial.

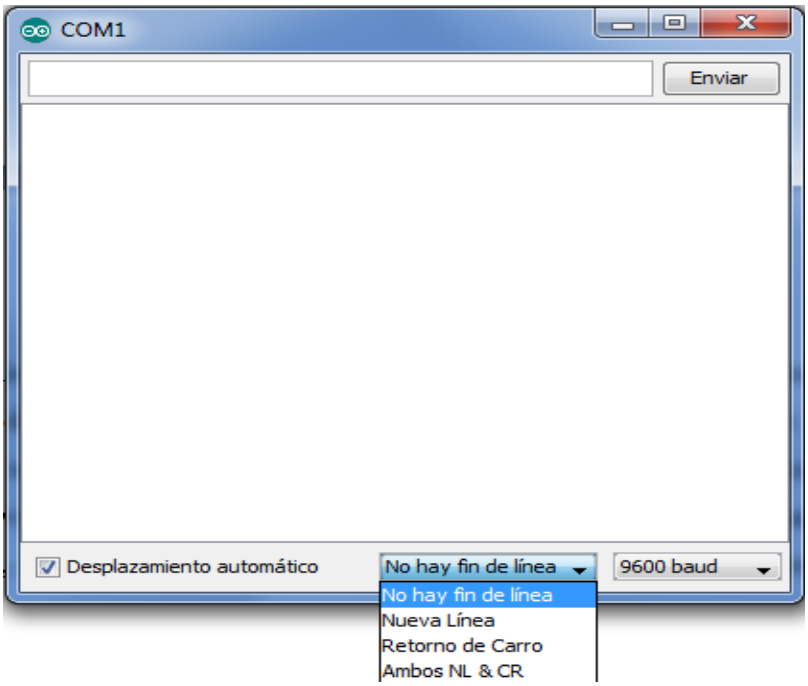
Nou: Per a la creació d'un nou sketch. Obre un nou full de text on escriurem el codi corresponent a l'sketch.

Obrir: Permet obrir un sketch ja existent que ha estat prèviament guardat. També pots obrir qualsevol dels esquetxos que porta instal·lats per defecte la IDE d'Arduino.

Guardar: Aquesta funcionalitat ens permet emmagatzemar el sketch que estem desenvolupant en aquest moment. Et permet triar la ruta en què vols guardar-la, i et crea automàticament una carpeta amb el mateix nom que li donis al sketch, guardant aquest dins de la mateixa.

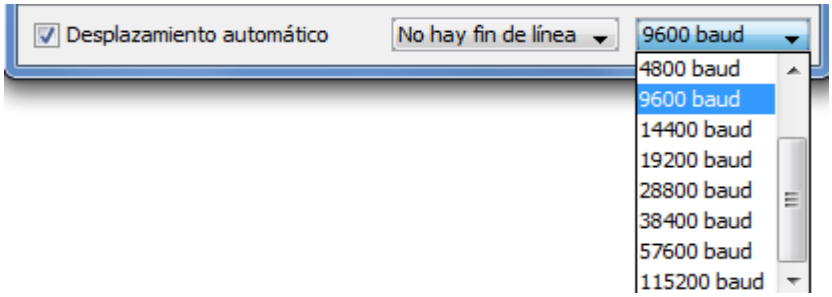
Monitor Serial: En fer clic sobre aquest icona, l'entorn de desenvolupament d'Arduino obre una nova finestra a través de la qual podem veure la comunicació establerta pel port sèrie entre la placa Arduino i el PC durant l'execució del programa. Conté una barra d'escriptura mitjançant la qual podem comunicar-nos amb Arduino a través del seu port sèrie, per exemple, per seleccionar diferents opcions que contemplin un possible menú creat per l'usuari dins d'un codi, o per enviar directament comandaments AT a una shield GSM/GPRS que tinguem muntada sobre l'Arduino.

A la següent il·lustració podem veure la pantalla corresponent al Monitor Serial i la pestanya desplegable en la qual podem seleccionar les diferents opcions referents als caràcters de final de línia.



Il·lustració 36: *Selecció de caràcters de fi de línia a la finestra "Monitor Serial"*

Dins el Monitor Serial disposem d'una altra pestanya per establir la taxa de bauds "baudrate", que marca el nombre d'unitats de senyal transmeses per segon. Aquest valor ha d'estar sincronitzat amb el baudrate en què estigui treballant l'Arduino, el qual pot ser establert en el codi del sketch mitjançant la comanda "Serial.begin" (valor del baudrate), o de no ser així, s'establirà un valor per defecte. Si el Monitor Serial i Arduino no estan sincronitzats amb la mateixa taxa de bauds, la informació que aparegui en la finestra serà completament il·legible. A la pròxima il·lustració apareix desplegada la pestanya per a la selecció dels diferents valors de baudrate disponibles.

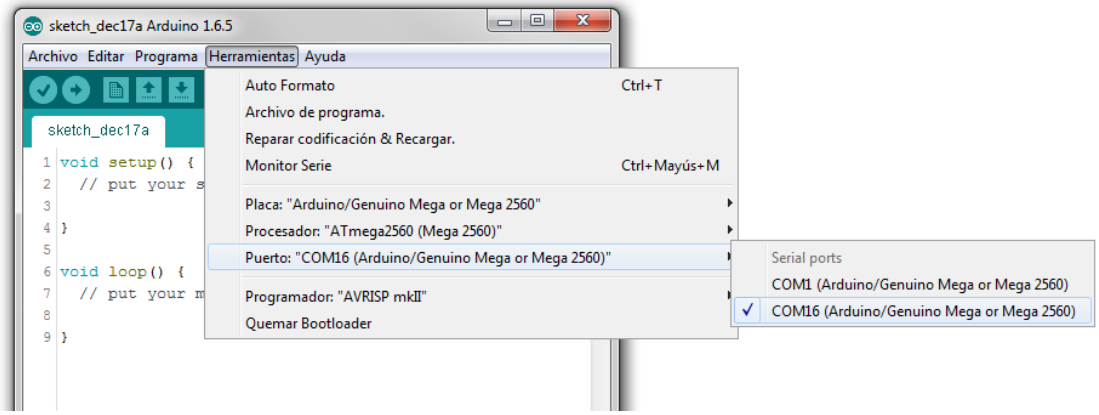


Il·lustració 37: *Selecció del valor de baudrate a la finestra "Monitor Serial"*

3.2 PARÀMETRES DE CONFIGURACIÓ DEL IDE PER ARDUINO

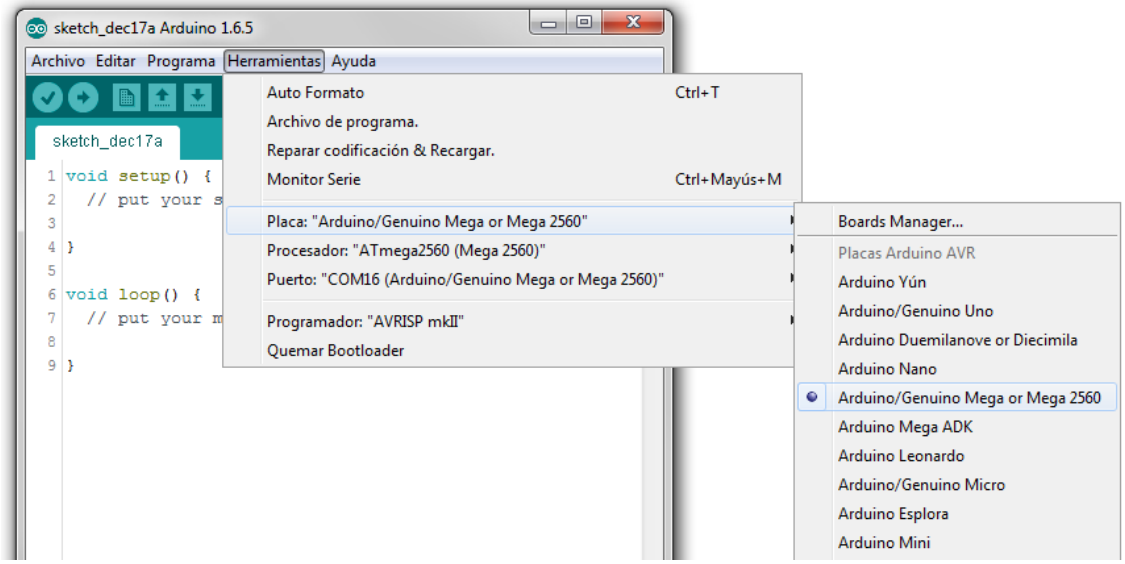
Per poder fer ús de les diferents funcions que ofereix Arduino, és indispensable haver instal·lat prèviament l'entorn de desenvolupament. A través del mateix, generarem els sketch amb el codi del programa que vulguem carregar al nostre dispositiu.

En primer lloc, descarreguem l'última versió de programari disponible a la mateixa web d'Arduino [16]. Pots descarregar la IDE acord amb el sistema operatiu que tinguis, Windows, Mac OS o Linux. Un cop descarregat, executem l'arxiu i seguim els passos de l'assistent d'instal·lació. Aquest pas no ha de suposar cap problema, la seva instal·lació és molt senzilla. Al Windows, el Arduino és detectat automàticament, però sempre se ha de confirmar. Tal com es mostra a la següent il·lustració, des del menú "Eines" podem seleccionar el port corresponent. Això sí, no oblidem tenir-lo connectat al port USB, de no ser així, no apareixerà el port i no podrem identificar-lo.



Il·lustració 38: *Selecció del port sèrie en el IDE de Arduino*

Un cop completada la instal·lació i identificat el port en què tenim connectat el nostre Arduino, arriba el torn de seleccionar el model concret de la nostra placa. Per a això accedim al menú "Eines", dins el mateix col·loquem el cursor sobre la pestanya "Board", i a continuació cliquem sobre la versió corresponent al nostre model, en el nostre cas, "Arduino Mega 2560". A la següent il·lustració podem veure el procediment.



Il·lustració 39: *Selecció model placa de Arduino*

Arribats a aquest punt, ja tenim instal·lat i configurat l'IDE d'Arduino. Per començar a treballar amb la nostra placa, només hem de generar un nou sketch i començar a escriure el codi de programació amb les instruccions que volem que executi el nostre Arduino.

3.3 INSTAL·LACIÓ I CONFIGURACIÓ DEL MÒDUL GSM/GPRS

Per fi entrem en unes de les parts més interessants del projecte, on treballarem amb els sistemes de comunicacions GSM/GPRS.

Recordem que el mòdul escollit per al nostre projecte va ser el "GSM/GPRS Quadband SIM900", el qual es mostra en la il·lustració següent.

Juntament amb el mòdul, s'ha d'adquirir una antena i una font d'alimentació externa, ja que els 500 mA d'Arduino poden no ser suficients per alimentar tant al mòdul com als components que connectem a la nostra plataforma. Vam adquirir també per tant els següents complements:

Antena:

- Freqüència: 900 MHz - 1800 MHz - 2.1 GHz
- Impedància: 50 Ohms
- Polarització: vertical
- Guany: 0 dBi
- VSWR: <2:1
- Potència: 25 W
- Connector: UFL
- Mesura: 8x1.2 cm
- Temperatura de funcionament: -40 °C a + 85 °C

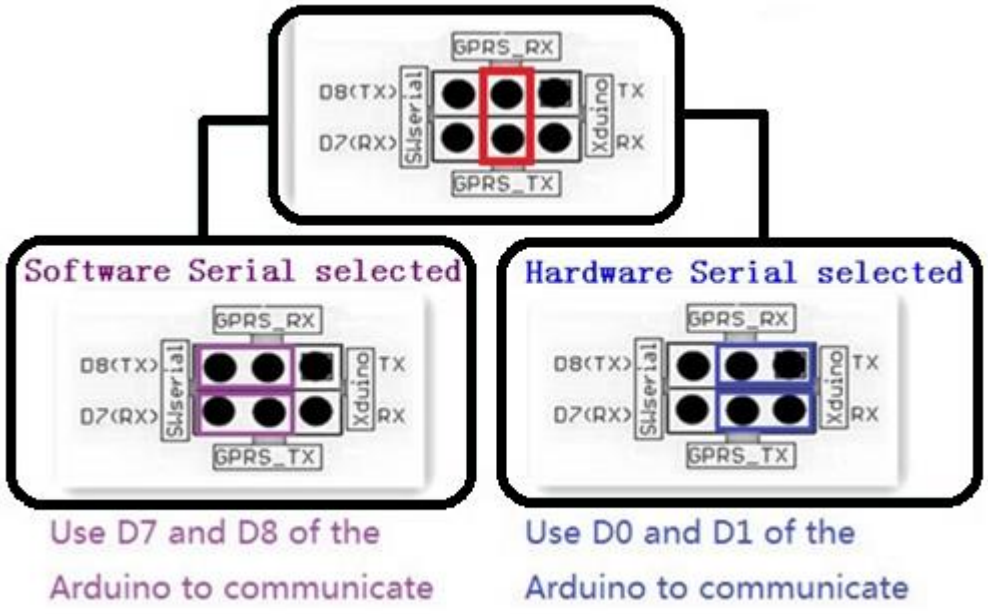
Font d'alimentació externa:

- Tensió d'entrada: 100 - 240 V
- Tensió de sortida: 7 V
- Corrent màxima: 1,2 A
- Diàmetre del connector: 2,1 mm
- Diàmetre de la coberta del connector: 5,5 mm

3.3.1 CONFIGURACIÓ PRÈVIA DEL MÒDUL

Ens disposem a provar el mòdul per això acudim a la pàgina tutorial del mateix [17], on podem trobar l'operativa recomanada a seguir per avaluar el seu correcte funcionament.

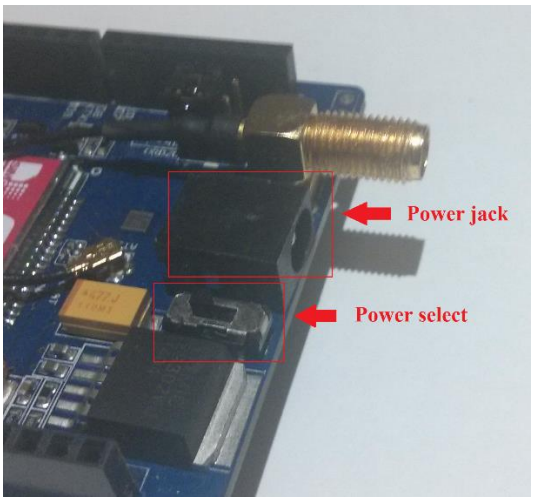
El primer apartat del manual d'aprenentatge fa referència al diagrama de connexions de la shield, el qual podem veure en detall en la següent il·lustració:



Il·lustració 40: *Diagrama de jumpers per el tipus de connexió amb la placa Arduino*

En el nostre cas elegim el mode "*Software Serial selected*" per provar el funcionament de les ordres AT.

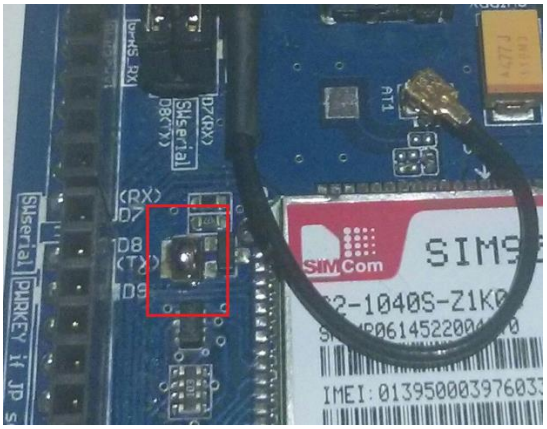
També amb de tenir present l'interruptor que determinarà la font d'alimentació del mòdul "*Power select*", si esta situat a la dreta del mòdul, s'alimentarà connectant el carregador de 7 V directament el mòdul "*Power jack*", però en el nostre cas el posicionarem a l'esquerra del mòdul, tal com apareix a la següent il·lustració, per alimentar-lo a traves de la placa Arduino ja que també necessitem un voltatge extra per els sensors meteorològics.



Il·lustració 41: *Interruptor d'alimentació del mòdul GSM/GPRS*

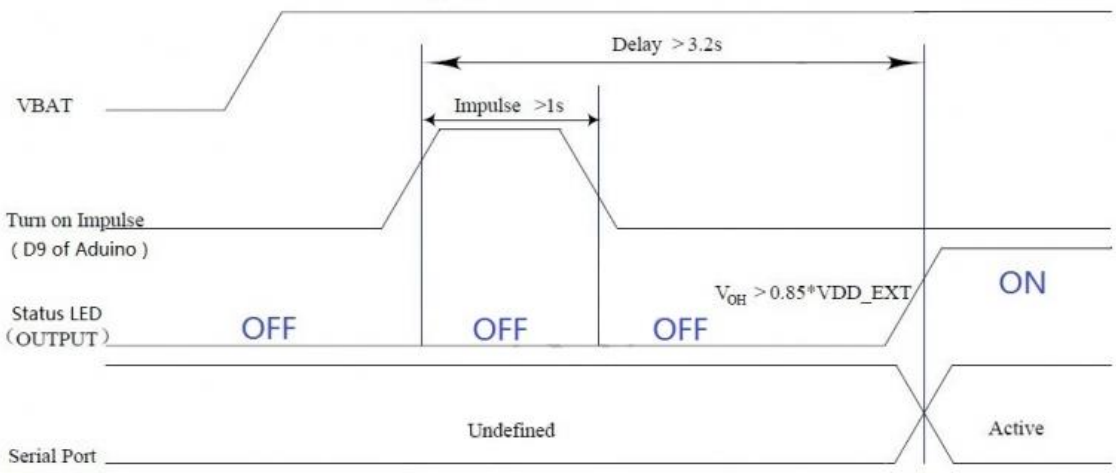
El mòdul es pot encendre de dues maneres, amb el botó ON/OFF o de forma automàtica enviant una ordre AT. Com que aquest projecte tracte de dissenyar una estació aïllada ens interessa automatitzar el màxim possible tots el processos i així minimitzar les intervencions presencials.

A continuació procedim a soldar els dos pins integrats del mòdul GSM/GPRS indicats en el manual per habilitar l'encès automàtic, l'únic inconvenient és que deixarà inhabilitat el pin d'entrada digital número 9, però en el nostre cas tenim pins d'entrada suficients.



Il·lustració 42: *Soldadura de pins encès automàtic mòdul GSM/GPRS*

En el següent esquema podem observar el voltatge i temps necessari el qual tenim que configurar la ordre AT, per encendre de forma automàtica el mòdul des de l'entrada digital número 9.

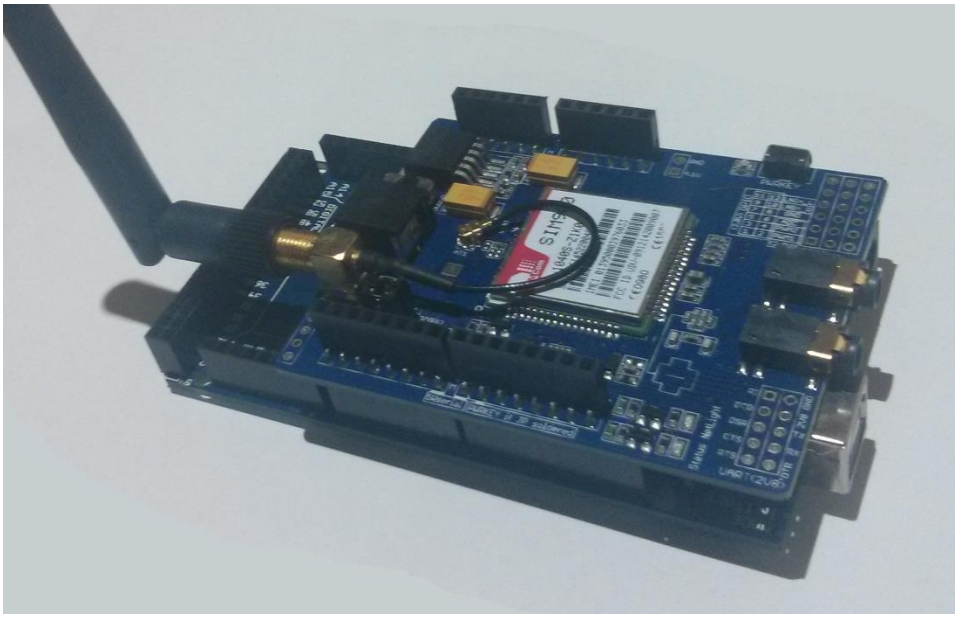


Il·lustració 43: *Esquema encès automàtic mòdul GSM/GPRS*

Per finalitzar és molt important tenir clars tots aquests conceptes referents als jumpers del mòdul, ja que serà el que més problemes pugui implicar a l'hora d'interactuar amb la placa. Un cop vist la connexió del mòdul, passem al muntatge de la mateixa sobre la placa Arduino MEGA.

Connectem l'antena GSM/GPRS el nostra mòdul, i introduïm la targeta SIM a la ranura situada el costat oposat del mòdul. Per a més comoditat, recomanem que lleveu prèviament el PIN de la vostra targeta SIM, així evitarem haver d'introduir una ordre o instrucció específica per a habilitar-la. Finalment, veiem el mòdul sobre la placa Arduino.

Il·lustració 44: *Targeta SIM en el mòdul GSM/GPRS*



Il·lustració 45: *Muntatge mòdul GSM/GPRS + Arduino MEGA*

3.3.2 ENVIAMENT DE SMS A TRAVÉS D'ARDUINO

Per poder comunicar-nos amb el mòdul GSM/GPRS a través d'Arduino i aconseguir enviar un SMS haurem de generar un codi compost per les següents ordres AT, de manera que quan carreguem aquest programa en la memòria del nostre Arduino aquest s'encarregui de transmetre aquests ordres el mòdul en el mateix ordre que ho faríem nosaltres manualment però de manera interna.

Els passos a seguir són:

1) Generar el sketch amb els comandaments AT adequats perquè el mòdul enviï un SMS amb la fase "Hola Mundo", adjuntem el codi del programa:

```
#include <Wire.h>
#include <SoftwareSerial.h>

int Led_placa = 13;
String Mensaje = "Hola Mundo";
String Numtelefono = "+34666666666";
SoftwareSerial SIM900(7,8);

void setup()
{
  Serial.begin(19200);
  SIM900.begin(19200);
  pinMode(Led_placa,OUTPUT);
  Encender_GSM();
  SMS_datos();
}

void loop()
{
}

void Encender_GSM()
{
  Serial.print("\nEncendiendo GSM");
  pinMode(9, OUTPUT);
  digitalWrite(9,LOW);
  delay(1000);
  digitalWrite(9,HIGH);
  delay(2000);
  digitalWrite(9,LOW);
  delay(3000);
  delay(30000);
  Serial.print("\nGSM Encendido\n");
}
```

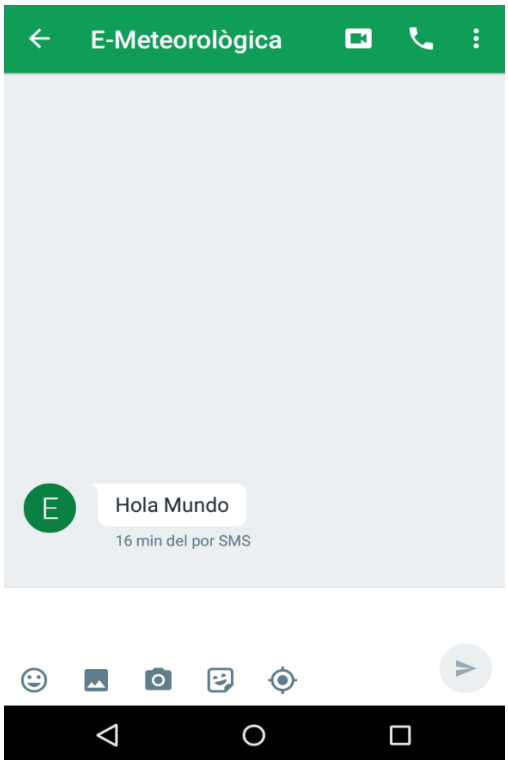
```
void SMS_datos()
{
  Serial.print("\nEnviando SMS");
  SIM900.print("AT+CMGF=1\r");
  delay(1000);
  SIM900.println("AT + CMGS = \""+Numtelefono+"\"");
  delay(1000);
  SIM900.print(Mensaje);
  delay(1000);
  SIM900.write((char)26);
  delay(1000);
  Serial.print("\nSMS Enviado\n");
}
```

2) Connectem la placa Arduino + mòdul GSM/GPRS el port USB de l'ordinador.

3) Carregem el sketch a la memòria de l'Arduino.

4) Connectem la font d'alimentació externa a la placar Arduino.

5) El programa comença a executar-se, i si tot ha anat bé, després d'uns segons rebrem el SMS al nostre mòbil, tal com es mostra a la següent il·lustració.



Il·lustració 46: *SMS enviat des de el mòdul GSM/GPRS*

3.3.3 LECTURA DE SMS A TRAVÉS D'ARDUINO

Entrem en l'última de les proves del mòdul GSM/GPRS que tenim intenció de fer abans d'endinsar-nos en el desenvolupament del programa general de la nostra aplicació. Es tracta de llegir un SMS que es troba a la targeta SIM a través d'un sketch carregat prèviament en la memòria del nostre Arduino.

El codi ve preparat per comunicar-se amb el mòdul alhora que tenim oberta la finestra corresponent al Monitor Serial, de manera que podem veure per pantalla les comandes que Arduino envia al mòdul i les respostes d'aquest, així com el contingut del missatge llegit.

1) Generar el sketch amb els comandaments AT adequats perquè el mòdul estigui preparat per rebre un SMS, i mostrar el seu contingut, adjuntem el codi del programa:

```
unsigned char buffer[64];
int count=0;

void setup()
{
  Serial1.begin(19200);
  Serial.begin(19200);
  Serial1.print("AT\r");
  delay(1000);
  leer();
  Serial1.print("AT+CMGF=1\r");
  delay(1000);
  leer();
  Serial1.print("AT+CPMS=\"SM\"\r");
  delay(1000);
  leer();
  delay(20000);
  Serial1.print("AT+CMGR=1\r");
  delay(1000);
  leer();
  Serial1.print("\nAT+CMGD=1\r");
  delay(1000);
  leer();
  Serial1.print("\nAT+CPMS=\"SM\"\r");
  delay(1000);
  leer();
}

void loop()
{}

void leer()
```

```
{
  if (Serial1.available())
  {
    while(Serial1.available())
    {
      buffer[count++]=Serial1.read();
      if(count == 64)break;
    }
    Serial.write(buffer,count);
    clearBufferArray();
    count = 0;
  }
  if (Serial.available())
    Serial1.write(Serial.read());
}

void clearBufferArray()
{
  for (int i=0; i<count;i++)
  {
    buffer[i]=NULL;
  }
}
```

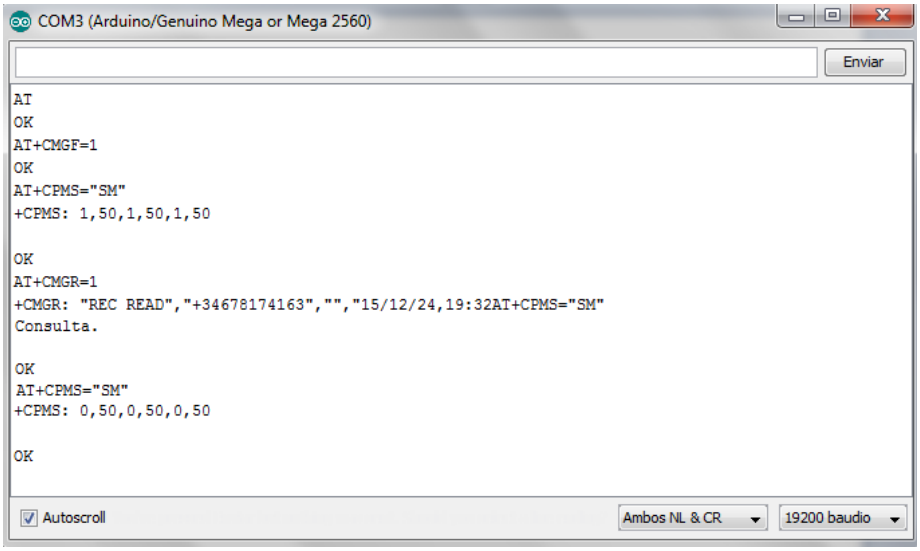
2) Connectem la placa Arduino + mòdul GSM/GPRS el port USB de l'ordinador.

3) Carregem el sketch a la memòria de l'Arduino.

4) Connectem la font d'alimentació externa a la placar Arduino.

5) El programa comença a executar-se, en aquets moments enviarem un SMS amb la paraula "Consulta." des de el nostre mòbil personal, i si tot ha anat bé, podrem visualitzar el contingut del SMS al Monitor Serial de l'Arduino, tal com es mostra a la següent il·lustració.

Il·lustració 47: *SMS enviat el mòdul GSM/GPRS*



Il·lustració 48: *Visualització del SMS rebut el mòdul GSM/GPRS en el Monitor Serial*

A continuació procedim a eliminar el SMS rebut, i comprova'm de nou que s'ha eliminat correctament i que la memòria de la SIM esta buida.

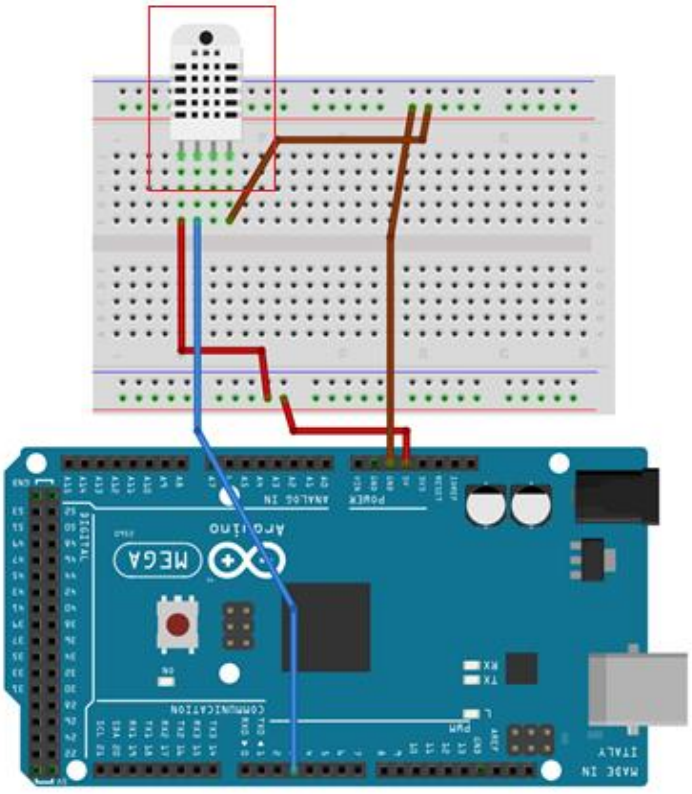
3.4 INSTAL·LACIÓ I CONFIGURACIÓ DELS SENSORS METEOROLÒGICS

En aquest apartat començarem a instal·lar tots i cada un del sensors de meteorologia anteriorment anomenats sobre una petita placa protoboard, per tal de realitzar les connexions i desenvolupar el codi de programació on cada un dels sensors actuaran sobre la placa Arduino.

3.4.1 SENSOR DE TEMPERATURA I HUMITAT

Començarem amb el sensor DHT22, que mesura la temperatura i humitat, aquestes mesures ens serviran per interpretar l'instrument meteorològic de Termòmetre i Psicròmetre.

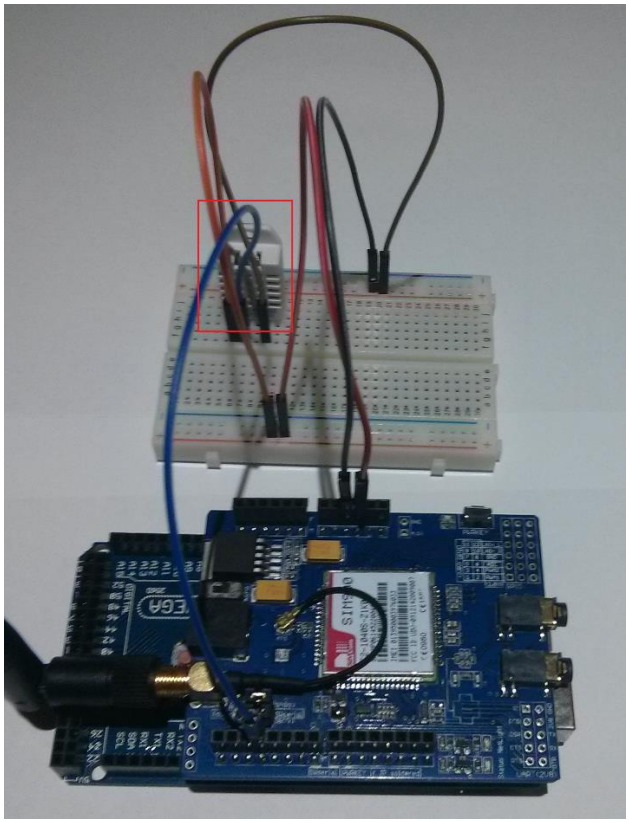
El primer que tenim que realitzar és l'esquema de muntatge corresponent, el resultat de la connexió amb la placa Arduino es mostra a la següent il·lustració.



Il·lustració 49: *Esquema connexió sensor Temperatura i Humitat*

Com podem observar a la il·lustració anterior del sensor DHT22, connectem la massa, el voltatge a 5 V, i el pin de sortida del sensor, esta connectat els pin d'entrada digital número 3 de la placa Arduino.

Procedim a realitzar el muntatge anterior a la placa Arduino més el mòdul GSM que havíem afegit anteriorment.



Il·lustració 50: *Muntatge mòdul GSM + sensor Temperatura i Humitat*

A continuació passem a generar un codi que obtingui el valor de la temperatura i humitat relativa, i ens ho enviï repetidament a través del Monitor Serial d'Arduino.

En aquest cas, a part del codi, també generarem la llibreria necessària per prendre el valors del sensor.

```
#include "DHT.h"

DHT::DHT(uint8_t pin, uint8_t type, uint8_t contar) {
  _pin = pin;
  _type = type;
  _contar = contar;
  inicio = true;
```

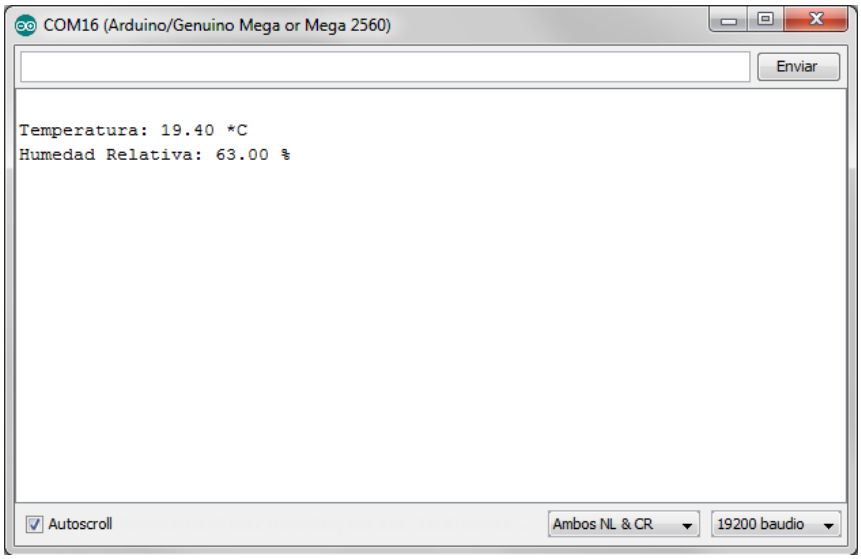
```
}

void DHT::begin(void) {
  pinMode(_pin, INPUT);
  digitalWrite(_pin, HIGH);
  _ultimo_tiempo_lectura = 0;
}

float DHT::Leer_temperatura(bool S) {
  float f;

  if (Leer()) {
    switch (_type) {
    case DHT11:
      f = Memoria[2];
      if(S)
        f = convertCtoF(f);

      return f;
    case DHT22:
    case DHT21:
      f = Memoria[2] & 0x7F;
      f *= 256;
      f += Memoria[3];
      f /= 10;
      if (Memoria[2] & 0x80)
  f *= -1;
      if(S)
  f = convertCtoF(f);

      return f;
    }
  }
  Serial.print("Fallo Lectura");
  return NAN;
}

float DHT::convertCtoF(float c) {
  return c * 9 / 5 + 32;
}

float DHT::readHumidity(void) {
  float f;
  if (Leer()) {
    switch (_type) {
    case DHT11:
      f = Memoria[0];
      return f;
    case DHT22:
    case DHT21:
      f = Memoria[0];
      f *= 256;
      f += Memoria[1];
      f /= 10;
      return f;
    }
  }
  Serial.print("Fallo Lectura");
  return NAN;
}
```

```
boolean DHT::read(void) {
  uint8_t Ultimo_estado = HIGH;
  uint8_t contador = 0;
  uint8_t j = 0, i;

  digitalWrite(_pin, HIGH);
  delay(250);

  currenttime = millis();
  if (currenttime < _ultimo_tiempo_lectura) {
    _ultimo_tiempo_lectura = 0;
  }
  if (!inicio && ((currenttime - _ultimo_tiempo_lectura) < 2000)) {
    return true
  }
  inicio = false;

  _ultimo_tiempo_lectura = millis();

  Memoria[0] = Memoria[1] = Memoria[2] = Memoria[3] = Memoria[4] = 0;

  pinMode(_pin, OUTPUT);
  digitalWrite(_pin, LOW);
  delay(20);
  digitalWrite(_pin, HIGH);
  delayMicroseconds(40);
  pinMode(_pin, INPUT);

  for ( i=0; i< MAXTIMINGS; i++) {
    contador = 0;
    while (digitalRead(_pin) == Ultimo_estado) {
      contador++;
      delayMicroseconds(1);
      if (contador == 255) {
        break;
      }
    }
    Ultimo_estado = digitalRead(_pin);

    if (contador == 255) break;

    if ((i >= 4) && (i%2 == 0)) {
      Memoria[j/8] <<= 1;
      if (contador > _contar)
        Memoria[j/8] |= 1;
      j++;
    }
  }
  if ((j >= 40) &&
      (Memoria[4] == ((Memoria[0] + Memoria[1] + Memoria[2] +
Memoria[3]) & 0xFF)) )
  {
    return true;
  }
  return false;
}
```

Una vegada que tenim la llibreria de referencia, només hem d'escriure el codi que s'encarregui d'enviar les dades a través del Monitor Serial d'Arduino.

```
#include <Wire.h>
#include <SoftwareSerial.h>
#include <DHT.h>

#define _SS_MAX_RX_BUFF 128
#define DHTPIN 3
#define DHTTYPE DHT22

int Led_placa = 13;
float t = 0;
float h = 0;
String Numtelefono1 = "+34666666666";
DHT dht(DHTPIN, DHTTYPE);
SoftwareSerial SIM900(7,8);

void setup()
{
  Serial.begin(19200);
  dht.begin();
  SIM900.begin(19200);
  pinMode(Led_placa,OUTPUT);
  t = dht.readTemperature();
  h = dht.readHumidity();
  Serial.print("\nTemperatura: ");
  Serial.print(t);
  Serial.print(" *C");
  Serial.print("\nHumedad Relativa: ");
  Serial.print(h);
  Serial.print(" %");
}

void loop()
{
  Serial.print("\nTemperatura: ");
  Serial.print(t);
  Serial.print(" *C");
  Serial.print("\nHumedad Relativa: ");
  Serial.print(h);
  Serial.print(" %");
  delay(5000);
  t = dht.readTemperature();
  h = dht.readHumidity();
}
```

Carregem el codi del sketch a l'Arduino, i podem observar el missatge amb les dades obtingudes en el Monitor Serial a la següent il·lustració.



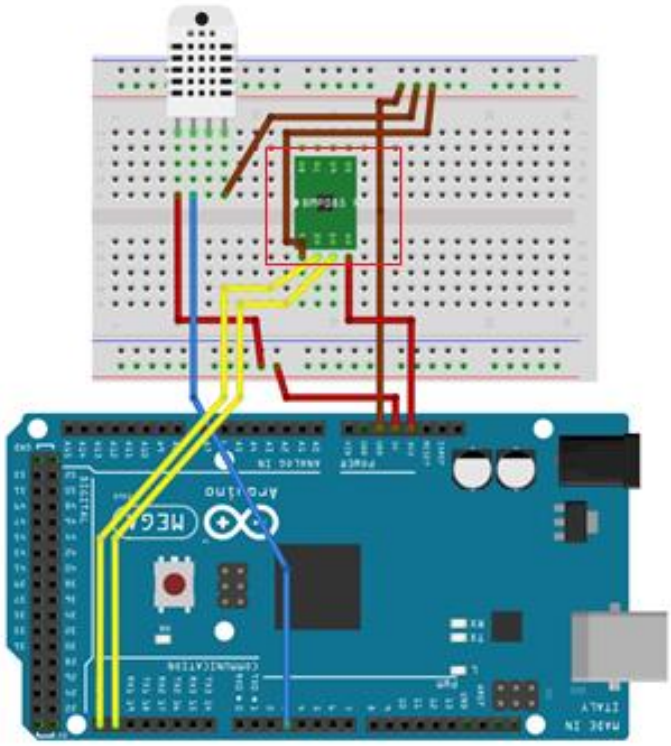
Il·lustració 51: *Dades de temperatura i humitat en el Monitor Serial*

En aquest cas el criteri de Temperatura te un valor de "19.40 ºC" i la humitat te un valor de "63.00 %".

3.4.2 SENSOR DE PRESSIÓ ATMOSFÈRICA I ALTITUD

El sensor BMP085, que mesura la pressió atmosfèrica i altitud, aquestes mesures ens serviran per interpretar l'instrument meteorològic de Psicròmetre i Altímetre.

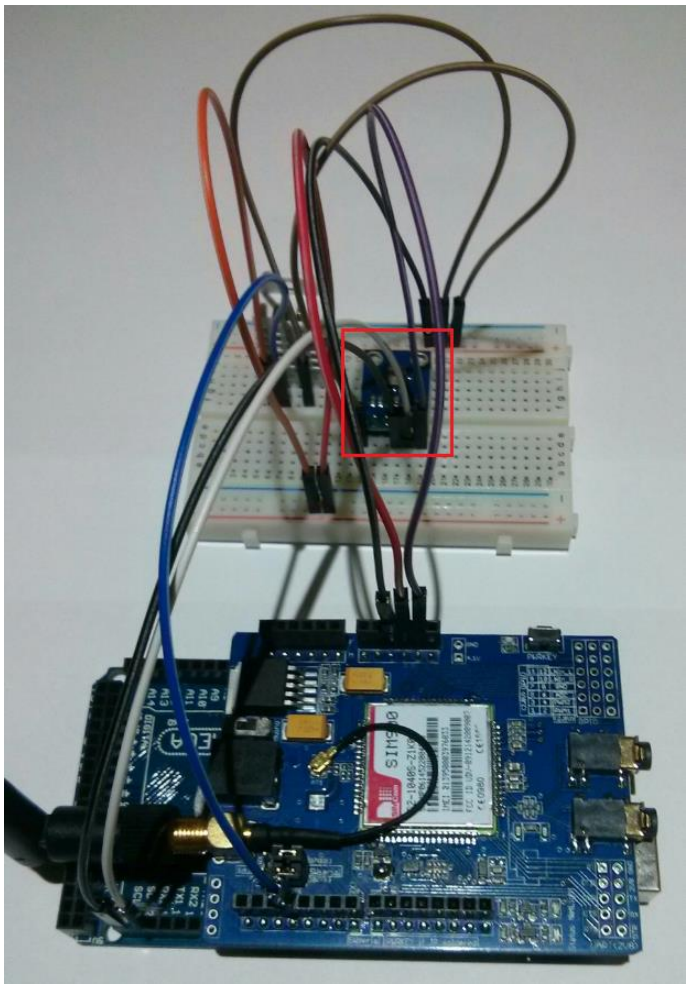
El primer que tenim que realitzar és l'esquema de muntatge corresponent, tangent especial atenció amb el voltatge d'entrada, així doncs com vàrem descriure anteriorment en les característiques d'aquest sensor, l'alimentació és de 3.3 V.
El resultat de la connexió amb la placa Arduino més el muntatge del sensor anterior es mostra a la següent il·lustració.



Il·lustració 52: *Esquema connexió sensor Temperatura i Humitat + sensor Pressió Atmosfèrica i Altitud*

Com podem observar a la il·lustració anterior, situem el sensor BMP085 al centre de la protoboard, connectant la massa, el voltatge a 3.3 V que ofereix directament un dels pins d'Arduino, i els dos pins de sortida del sensor, estan connectats els pins de comunicació SDA i SCL de la placa Arduino.

Procedim a realitzar el muntatge anterior a la placa Arduino més el mòdul GSM i el sensor que havíem afegit anteriorment.



Il·lustració 53: *Muntatge mòdul GSM + sensor Temperatura i Humitat + sensor Pressió Atmosfèrica i Altitud*

A continuació passem a generar un codi que obtingui el valor de la Pressió Atmosfèrica i Altitud, i ens ho enviï repetidament a través del Monitor Serial d'Arduino.
En aquest cas, a part del codi, també generarem la llibreria necessària per prendre el valors del sensor.

```
#include "BMP085.h"
#include <util/delay.h>
BMP085::BMP085() {}

void BMP085::begin(uint8_t mode) {
  if (mode > BMP085_ULTRAHIGHRES)
    mode = BMP085_ULTRAHIGHRES;
  oversampling = mode;
  Wire.begin();

  ac1 = read16(BMP085_CAL_AC1);
  ac2 = read16(BMP085_CAL_AC2);
  ac3 = read16(BMP085_CAL_AC3);
  ac4 = read16(BMP085_CAL_AC4);
  ac5 = read16(BMP085_CAL_AC5);
  ac6 = read16(BMP085_CAL_AC6);
  b1 = read16(BMP085_CAL_B1);
  b2 = read16(BMP085_CAL_B2);
  mb = read16(BMP085_CAL_MB);
  mc = read16(BMP085_CAL_MC);
  md = read16(BMP085_CAL_MD);
#if (BMP085_DEBUG == 1)
  Serial.print("ac1 = "); Serial.println(ac1, DEC);
  Serial.print("ac2 = "); Serial.println(ac2, DEC);
  Serial.print("ac3 = "); Serial.println(ac3, DEC);
  Serial.print("ac4 = "); Serial.println(ac4, DEC);
  Serial.print("ac5 = "); Serial.println(ac5, DEC);
  Serial.print("ac6 = "); Serial.println(ac6, DEC);
  Serial.print("b1 = "); Serial.println(b1, DEC);
  Serial.print("b2 = "); Serial.println(b2, DEC);
  Serial.print("mb = "); Serial.println(mb, DEC);
  Serial.print("mc = "); Serial.println(mc, DEC);
  Serial.print("md = "); Serial.println(md, DEC);
#endif
}
uint16_t BMP085::leerRawTemperatura(void) {
  write8(BMP085_CONTROL, BMP085_READTEMPCMD);
  _delay_ms(5);
#if BMP085_DEBUG == 1
  Serial.print("Raw temp: "); Serial.println(read16(BMP085_TEMPDATA));
#endif
  return read16(BMP085_TEMPDATA);
}
uint32_t BMP085::leerRawPresion(void) {
  uint32_t raw;
  write8(BMP085_CONTROL, BMP085_LeerPresionCMD + (oversampling << 6));
  if (oversampling == BMP085_ULTRALOWPOWER)
    _delay_ms(5);
  else if (oversampling == BMP085_STANDARD)
    _delay_ms(8);
  else if (oversampling == BMP085_HIGHRES)
    _delay_ms(14);
  else
    _delay_ms(26);
  raw = read16(BMP085_DatosPresion);
  raw <<= 8;
  raw |= read8(BMP085_DatosPresion+2);
  raw >>= (8 - oversampling);

#if BMP085_DEBUG == 1
  Serial.print("Raw presion: "); Serial.println(raw);
```

```
#endif
  return raw;
}
int32_t BMP085::LeerPresion(void) {
  int32_t UT, UP, B3, B5, B6, X1, X2, X3, p;
  uint32_t B4, B7;
  UT = leerRawTemperatura();
  UP = leerRawPresion();
#if BMP085_DEBUG == 1
  UT = 27898;
  UP = 23843;
  ac6 = 23153;
  ac5 = 32757;
  mc = -8711;
  md = 2868;
  b1 = 6190;
  b2 = 4;
  ac3 = -14383;
  ac2 = -72;
  ac1 = 408;
  ac4 = 32741;
  oversampling = 0;
#endif
  X1 = ((UT - (int32_t)ac6) * (int32_t)ac5) >> 15;
  X2 = ((int32_t)mc << 11) - (X1 + md)/2;
  X2 /= (X1 + md);
  B5 = X1 + X2;
#if BMP085_DEBUG == 1
  Serial.print("X1 = "); Serial.println(X1);
  Serial.print("X2 = "); Serial.println(X2);
  Serial.print("B5 = "); Serial.println(B5);
#endif
  B6 = B5 - 4000;
  X1 = ((int32_t)b2 * ( (B6 * B6)>>12 )) >> 11;
  X2 = ((int32_t)ac2 * B6) >> 11;
  X3 = X1 + X2;
  B3 = ((((int32_t)ac1*4 + X3) << oversampling) + 2) / 4;
#if BMP085_DEBUG == 1
  Serial.print("B6 = "); Serial.println(B6);
  Serial.print("X1 = "); Serial.println(X1);
  Serial.print("X2 = "); Serial.println(X2);
  Serial.print("B3 = "); Serial.println(B3);
#endif
  X1 = ((int32_t)ac3 * B6) >> 13;
  X2 = ((int32_t)b1 * ((B6 * B6) >> 12)) >> 16;
  X3 = ((X1 + X2) + 2) >> 2;
  B4 = ((uint32_t)ac4 * (uint32_t)(X3 + 32768)) >> 15;
  B7 = ((uint32_t)UP - B3) * (uint32_t)( 50000UL >> oversampling );
#if BMP085_DEBUG == 1
  Serial.print("X1 = "); Serial.println(X1);
  Serial.print("X2 = "); Serial.println(X2);
  Serial.print("B4 = "); Serial.println(B4);
  Serial.print("B7 = "); Serial.println(B7);
#endif
  if (B7 < 0x80000000) {
    p = (B7 * 2) / B4;
  } else {
    p = (B7 / B4) * 2;
  }
  X1 = (p >> 8) * (p >> 8);
  X1 = (X1 * 3038) >> 16;
```

```
  X2 = (-7357 * p) >> 16;
#if BMP085_DEBUG == 1
  Serial.print("p = "); Serial.println(p);
  Serial.print("X1 = "); Serial.println(X1);
  Serial.print("X2 = "); Serial.println(X2);
#endif
  p = p + ((X1 + X2 + (int32_t)3791)>>4);
#if BMP085_DEBUG == 1
  Serial.print("p = "); Serial.println(p);
#endif
  return p;
}
float BMP085::readTemperature(void) {
  int32_t UT, X1, X2, B5;
  float temp;

  UT = leerRawTemperatura();

#if BMP085_DEBUG == 1
  UT = 27898;
  ac6 = 23153;
  ac5 = 32757;
  mc = -8711;
  md = 2868;
#endif
  X1 = ((UT - (int32_t)ac6) * (int32_t)ac5) >> 15;
  X2 = ((int32_t)mc << 11) / (X1 + (int32_t)md);
  B5 = X1 + X2;
  temp = (B5 + 8) >> 4;
  temp /= 10;
  return temp;
}
float BMP085::LeerAltitud(float Presion_nivel_mar) {
  float altitud;
  float presion = LeerPresion();
  altitud = 44330 * (1.0 - pow(presion /Presion_nivel_mar,0.1903));
  return altitud;
}
uint8_t BMP085::read8(uint8_t a) {
  uint8_t ret;
  Wire.beginTransmission(BMP085_I2CADDR);
#if (ARDUINO >= 100)
  Wire.write(a);
#else
  Wire.send(a);
#endif
  Wire.endTransmission();

  Wire.beginTransmission(BMP085_I2CADDR);
  Wire.requestFrom(BMP085_I2CADDR, 1);
#if (ARDUINO >= 100)
  ret = Wire.read();
#else
  ret = Wire.receive();
#endif
  Wire.endTransmission();
  return ret;
}
uint16_t BMP085::read16(uint8_t a) {
  uint16_t ret;
  Wire.beginTransmission(BMP085_I2CADDR);
```

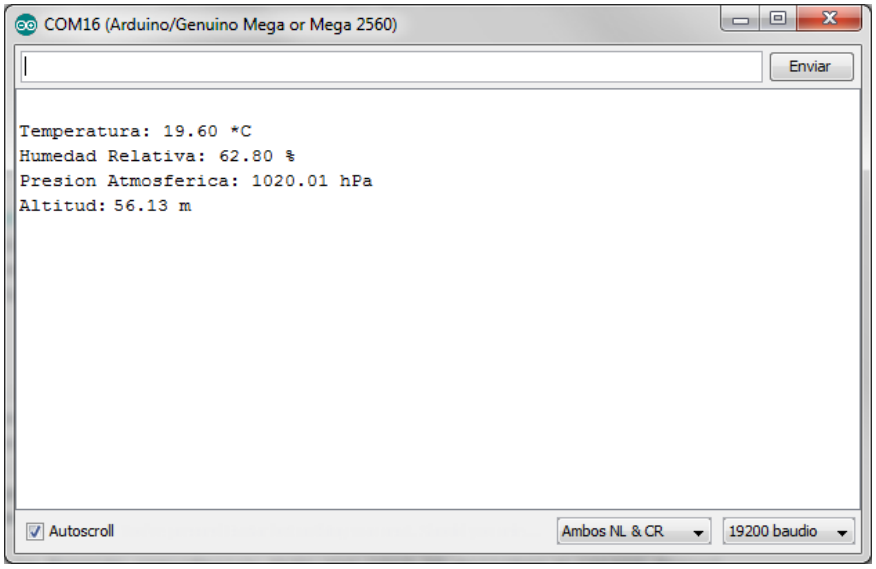
```
#if (ARDUINO >= 100)
  Wire.write(a);
#else
  Wire.send(a);
#endif
  Wire.endTransmission();
  Wire.beginTransmission(BMP085_I2CADDR);
  Wire.requestFrom(BMP085_I2CADDR, 2);
#if (ARDUINO >= 100)
  ret = Wire.read();
  ret <<= 8;
  ret |= Wire.read();
#else
  ret = Wire.receive();
  ret <<= 8;
  ret |= Wire.receive();
#endif
  Wire.endTransmission();
  return ret;
}
void BMP085::write8(uint8_t a, uint8_t d) {
  Wire.beginTransmission(BMP085_I2CADDR);
#if (ARDUINO >= 100)
  Wire.write(a);
  Wire.write(d);
#else
  Wire.send(a);
  Wire.send(d);
#endif
  Wire.endTransmission();
}
```

Una vegada que tenim la llibreria de referencia, només hem d'escriure el codi que s'encarregui d'enviar les dades a través del Monitor Serial d'Arduino.

```
#include <Wire.h>
#include <SoftwareSerial.h>
#include <DHT.h>
#include <BMP085.h>

#define _SS_MAX_RX_BUFF 128
#define DHTPIN 3
#define DHTTYPE DHT22

int Led_placa = 13;
int Led = 4;
float t = 0;
float h = 0;
float p = 0;
float a = 0;
String Numtelefono1 = "+34666666666";
DHT dht(DHTPIN, DHTTYPE);
BMP085 bmp;
SoftwareSerial SIM900(7,8);

void setup()
```

```
{
  Serial.begin(19200);
  dht.begin();
  bmp.begin();
  SIM900.begin(19200);
  pinMode(Led_placa,OUTPUT);
  t = dht.readTemperature();
  h = dht.readHumidity();
  p = bmp.readPressure();
  a = bmp.readAltitude();
  Serial.print("\nTemperatura: ");
  Serial.print(t);
  Serial.print(" *C");
  Serial.print("\nHumedad Relativa: ");
  Serial.print(h);
  Serial.print(" %");
  Serial.print("\nPresion Atmosferica: ");
  Serial.print(p/100);
  Serial.print(" hPa");
  Serial.print("\nAltitud: ");
  Serial.print(a);
  Serial.print(" m");
}

void loop()
{
  Serial.print("\nTemperatura: ");
  Serial.print(t);
  Serial.print(" *C");
  Serial.print("\nHumedad Relativa: ");
  Serial.print(h);
  Serial.print(" %");
  Serial.print("\nPresion Atmosferica: ");
  Serial.print(p/100);
  Serial.print(" hPa");
  Serial.print("\nAltitud: ");
  Serial.print(a);
  Serial.print(" m");
  delay(5000);
  t = dht.readTemperature();
  h = dht.readHumidity();
  p = bmp.readPressure();
  a = bmp.readAltitude();
}
```

Carregem el codi del sketch a l'Arduino, i podem observar el missatge amb les dades obtingudes en el Monitor Serial a la següent il·lustració.



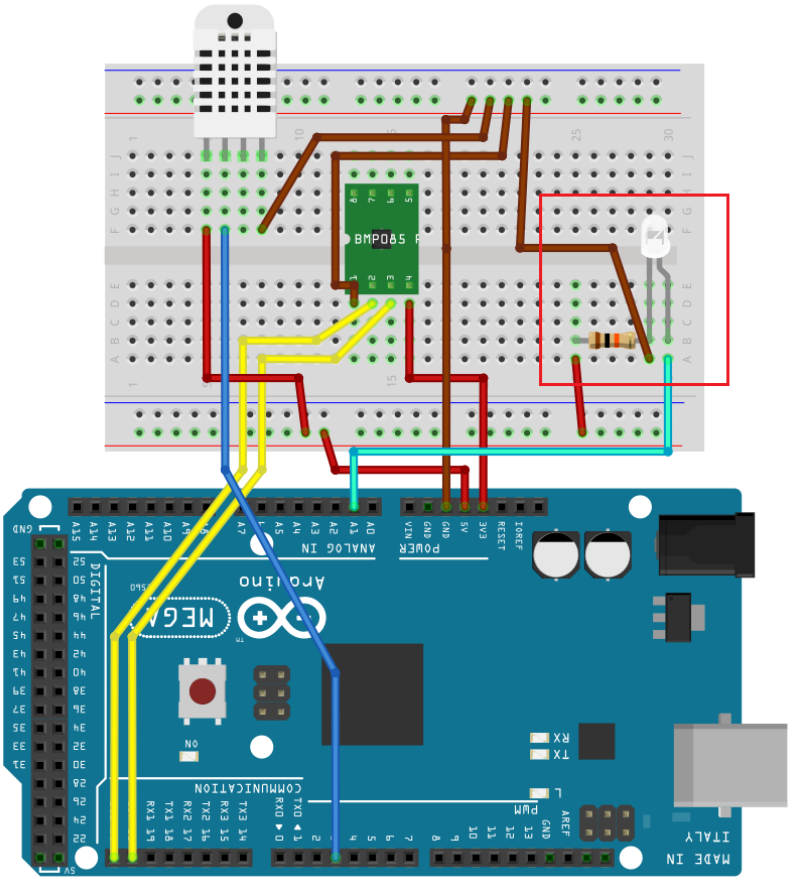
Il·lustració 54: *Dades de Temperatura , Humitat, Pressió Atmosfèrica i Altitud en el Monitor Serial*

En aquest cas el criteri de Pressió Atmosfèrica te un valor de "1020.01 hPa" i l'Altitud te un valor de "56.13 m".

3.4.3 SENSOR DE LLUMINOSITAT

Per aquest sensor utilitzarem un fotodíode, que mesura el nivell d'intensitat de la llum, aquest sensor ens servirà per interpretar l'instrument meteorològic Heliògraf.

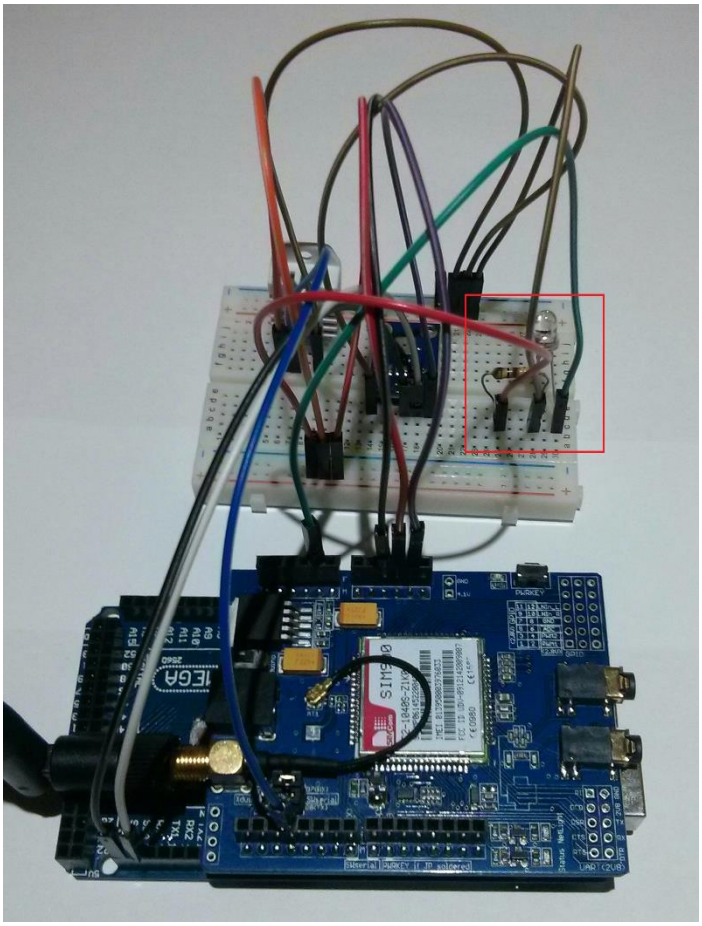
El primer que tenim que realitzar és l'esquema de muntatge corresponent, és recomanable col·locar una resistència de 10KΩ entre la massa i l'extrem negatiu del fotodíode, l'altre patilla la connectem directament en el pin digital de l'Arduino.
El resultat de la connexió amb la placa Arduino més el muntatge del sensors anteriors es mostra a la següent il·lustració.



Il·lustració 55: *Esquema connexió sensor Temperatura i Humitat + sensor Pressió Atmosfèrica i Altitud + sensor lluminositat*

Com podem observar a la il·lustració anterior del fotodíode, connectem la patilla de l'ànode junt amb la massa i la resistència de 10KΩ, la patilla del càtode està connectat el pin d'entrada analògica número 1 de la placa Arduino, i la patilla positiva de la resistència connectada el voltatge d'entrada de 5 V.

Procedim a realitzar el muntatge anterior a la placa Arduino més el mòdul GSM i els sensors que havíem afegit anteriorment.



Il·lustració 56: *Muntatge mòdul GSM + sensor Temperatura i Humitat + sensor Pressió Atmosfèrica i Altitud + sensor lluminositat*

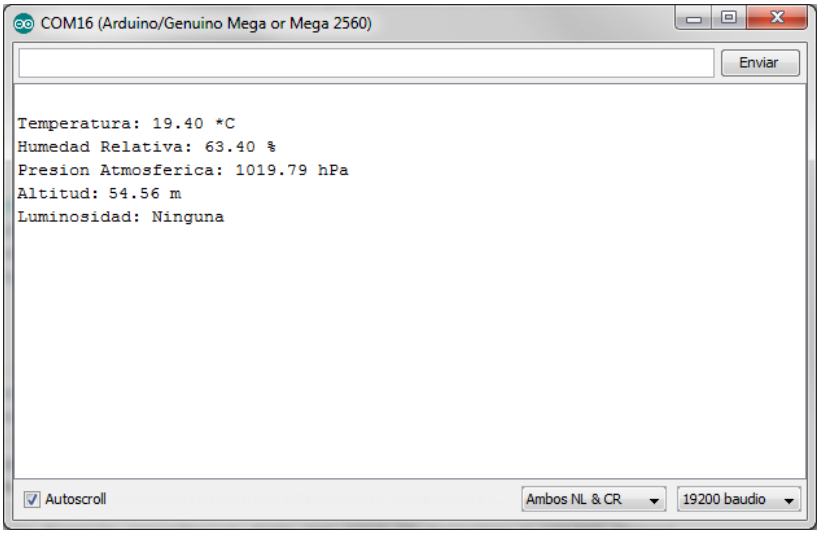
A continuació passem a generar el codi que obtingui el valor de la lluminositat, i ens ho enviï repetidament a través del Monitor Serial d'Arduino.

```
#include <Wire.h>
#include <SoftwareSerial.h>
#include <DHT.h>
#include <BMP085.h>

#define _SS_MAX_RX_BUFF 128
#define DHTPIN 3
#define DHTTYPE DHT22

int Led_placa = 13;
```

```
float t = 0;
float h = 0;
float p = 0;
float a = 0;
int lm = 0;
String Numtelefono1 = "+34666666666";
DHT dht(DHTPIN, DHTTYPE);
BMP085 bmp;
SoftwareSerial SIM900(7,8);

void setup()
{
  Serial.begin(19200);
  dht.begin();
  bmp.begin();
  SIM900.begin(19200);
  pinMode(Led_placa,OUTPUT);
  t = dht.readTemperature();
  h = dht.readHumidity();
  p = bmp.readPressure();
  a = bmp.readAltitude();
  lm = analogRead(A1);
  Serial.print("\nTemperatura: ");
  Serial.print(t);
  Serial.print(" *C");
  Serial.print("\nHumedad Relativa: ");
  Serial.print(h);
  Serial.print(" %");
  Serial.print("\nPresion Atmosferica: ");
  Serial.print(p/100);
  Serial.print(" hPa");
  Serial.print("\nAltitud: ");
  Serial.print(a);
  Serial.print(" m");
  if (lm >= 90)
    {Serial.print("\nLuminosidad: Alta");}
    else if (lm >= 80)
    {Serial.print("\nLuminosidad: Media");}
    else if (lm >= 70)
    {Serial.print("\nLuminosidad: Baja");}
    else
    {Serial.print("\nLuminosidad: Ninguna");}
}

void loop()
{
  Serial.print("\nTemperatura: ");
  Serial.print(t);
  Serial.print(" *C");
  Serial.print("\nHumedad Relativa: ");
  Serial.print(h);
  Serial.print(" %");
  Serial.print("\nPresion Atmosferica: ");
  Serial.print(p/100);
  Serial.print(" hPa");
  Serial.print("\nAltitud: ");
  Serial.print(a);
  Serial.print(" m");
  if (lm >= 90)
    {Serial.print("\nLuminosidad: Alta");}
    else if (lm >= 80)
```

```
    {Serial.print("\nLuminosidad: Media");}
    else if (lm >= 70)
    {Serial.print("\nLuminosidad: Baja");}
    else
    {Serial.print("\nLuminosidad: Ninguna");}
  delay(10000);
  t = dht.readTemperature();
  h = dht.readHumidity();
  p = bmp.readPressure();
  a = bmp.readAltitude();
  lm = analogRead(A1);
}
```

Carregem el codi del sketch a l'Arduino, i podem observar el missatge amb les dades obtingudes en el Monitor Serial a la següent il·lustració.



Il·lustració 57: *Dades de Temperatura , Humitat, Pressió Atmosfèrica, Altitud i lluminositat en el Monitor Serial*

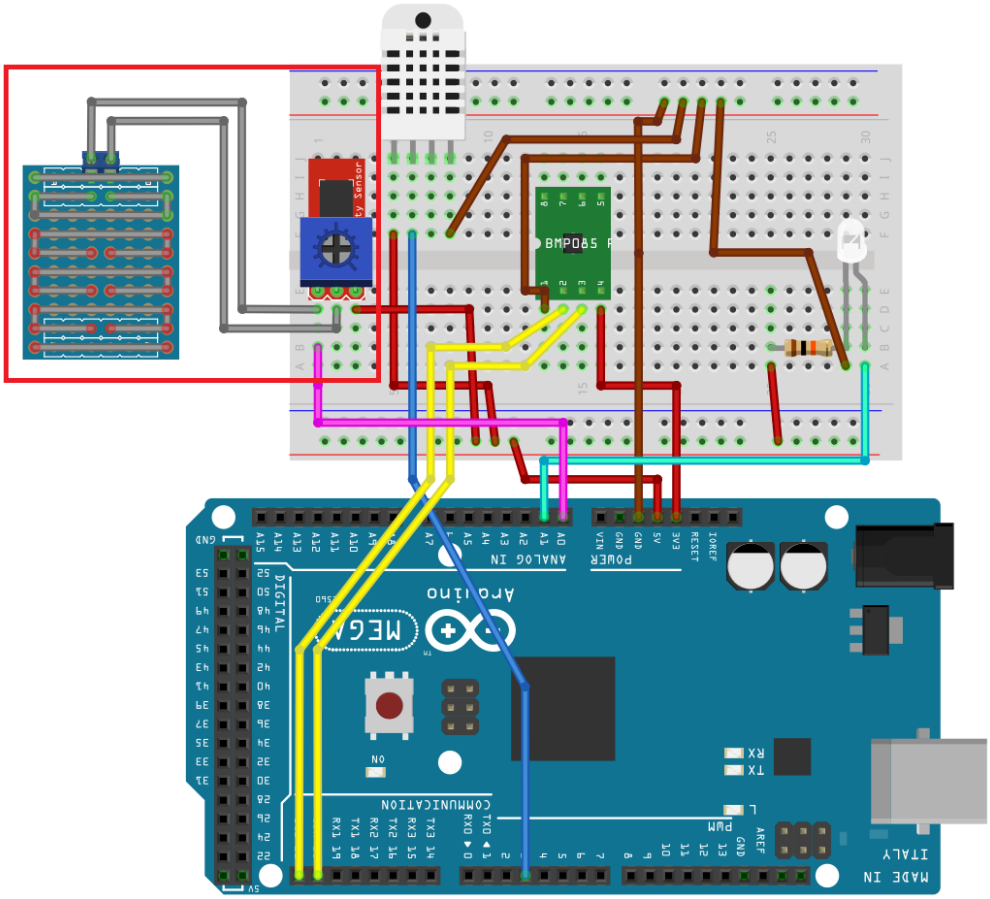
En aquest cas el criteri de Lluminositat te un valor de "Ninguna" ja que estem a l'interior d'una vivenda amb poca llum artificial, però pot variar entre: Alta, Mitjana, Baixa o Ninguna.

3.4.4 SENSOR DE PLUJA

Per aquest sensor utilitzarem el model EL0405, no mesura la quantitat d'aigua que cau en un metre quadrat de superfície, el qual interpretaria l'instrument meteorològic Pluviòmetre, si no que ens informa si esta plovent o no.

El primer que tenim que realitzar és l'esquema de muntatge corresponent, el sensor de pluja esta formada per dues parts, una placa resistiva que detecta les gotes de pluja quant l'aigua entrar en contacte amb la placa i es produeix una circulació de corrent, esta fabricada amb materials especialment antioxidants. L'altre part es situa el controlador EL0405 acompanyat d'un potenciòmetre que ens permet ajustar la sensibilitat de la senyal que s'envia a l'Arduino.

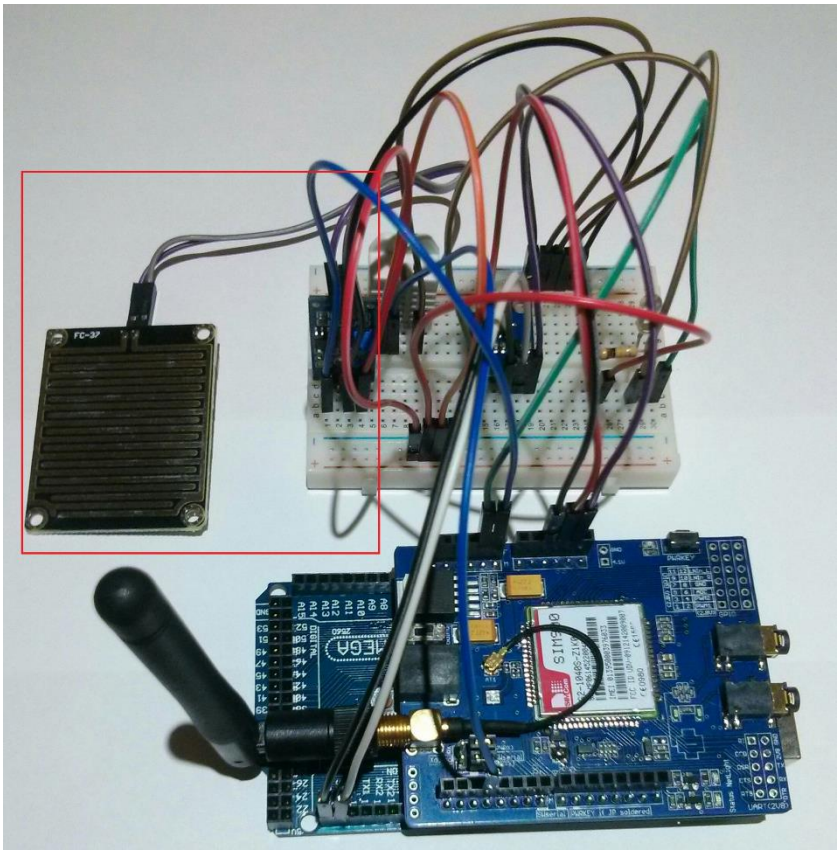
El resultat de la connexió amb la placa Arduino més el muntatge del sensors anteriors es mostra a la següent il·lustració.



Il·lustració 58: *Esquema connexió sensor Temperatura i Humitat + sensor Pressió Atmosfèrica i Altitud + sensor lluminositat + sensor de pluja*

Com podem observar a la il·lustració anterior, es connecten les dues patilles de la placa resistiva el controlador del sensor de pluja, també connectem una de les patilles a massa i l'altre el voltatge de 5 V. Finalment la patilla de la senyal de sortida del sensor el connectem el pin d'entrada analògica número 0 de la placa Arduino.

Procedim a realitzar el muntatge anterior a la placa Arduino més el mòdul GSM i els sensors que havíem afegit anteriorment.



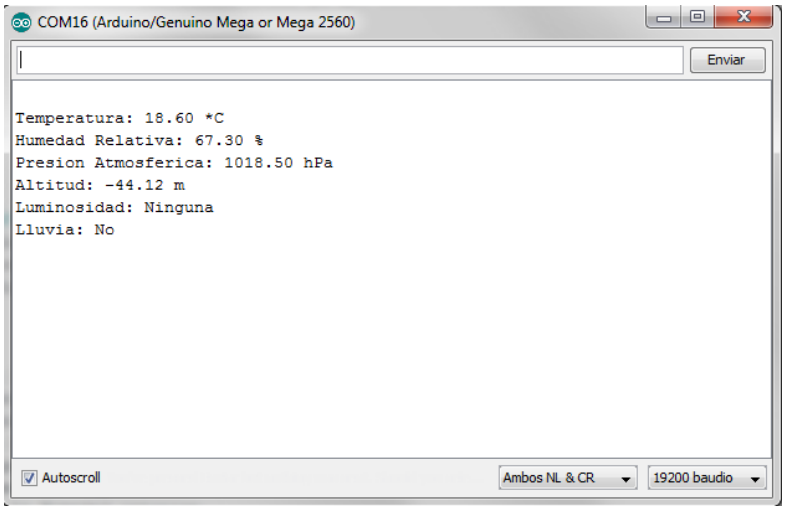
Il·lustració 59: *Muntatge mòdul GSM + sensor Temperatura i Humitat + sensor Pressió Atmosfèrica i Altitud + sensor lluminositat + sensor de pluja*

A continuació passem a generar el codi que obtingui el valor de resistivitat del sensor quant es banya, per determinar si esta plovent o no, i ens ho enviï repetidament a través del Monitor Serial d'Arduino.

```
#include <Wire.h>
#include <SoftwareSerial.h>
#include <DHT.h>
#include <BMP085.h>

#define _SS_MAX_RX_BUFF 128
#define DHTPIN 3
#define DHTTYPE DHT22

int Led_placa = 13;
float t = 0;
float h = 0;
float p = 0;
float a = 0;
int lm = 0;
int llv = 0;
String Numtelefono1 = "+34666666666";
DHT dht(DHTPIN, DHTTYPE);
BMP085 bmp;
SoftwareSerial SIM900(7,8);

void setup()
{
  Serial.begin(19200);
  dht.begin();
  bmp.begin();
  SIM900.begin(19200);
  pinMode(Led_placa,OUTPUT);
  t = dht.readTemperature();
  h = dht.readHumidity();
  p = bmp.readPressure();
  a = bmp.readAltitude();
  llv = analogRead(A0);
  lm = analogRead(A1);
}

void loop()
{
  Serial.print("\nTemperatura: ");
  Serial.print(t);
  Serial.print(" *C");
  Serial.print("\nHumedad Relativa: ");
  Serial.print(h);
  Serial.print(" %");
  Serial.print("\nPresion Atmosferica: ");
  Serial.print(p/100);
  Serial.print(" hPa");
  Serial.print("\nAltitud: ");
  Serial.print(a);
  Serial.print(" m");
  if (lm >= 90)
    {Serial.print("\nLuminosidad: Alta");}
    else if (lm >= 80)
    {Serial.print("\nLuminosidad: Media");}
    else if (lm >= 70)
    {Serial.print("\nLuminosidad: Baja");}
    else
    {Serial.print("\nLuminosidad: Ninguna");}
  if (llv <= 270)
    {Serial.print("\nLluvia: Si\n");}
    else
```

```
    {Serial.print("\nLluvia: No\n");}
  delay(5000);
  t = dht.readTemperature();
  h = dht.readHumidity();
  p = bmp.readPressure();
  a = bmp.readAltitude();
  llv = analogRead(A0);
  lm = analogRead(A1);
}
```

Carregem el codi del sketch a l'Arduino, i podem observar el missatge amb les dades obtingudes en el Monitor Serial a la següent il·lustració.



Il·lustració 60: *Dades de Temperatura , Humitat, Pressió Atmosfèrica, Altitud, lluminositat i pluja en el Monitor Serial*

En aquest cas el criteri de Pluja te un valor de "No" ja que estem a l'interior d'una vivenda, però pot variar entre: Si o No, depenent si la placa resistiva que detecta l'aigua, es banya quant plou.

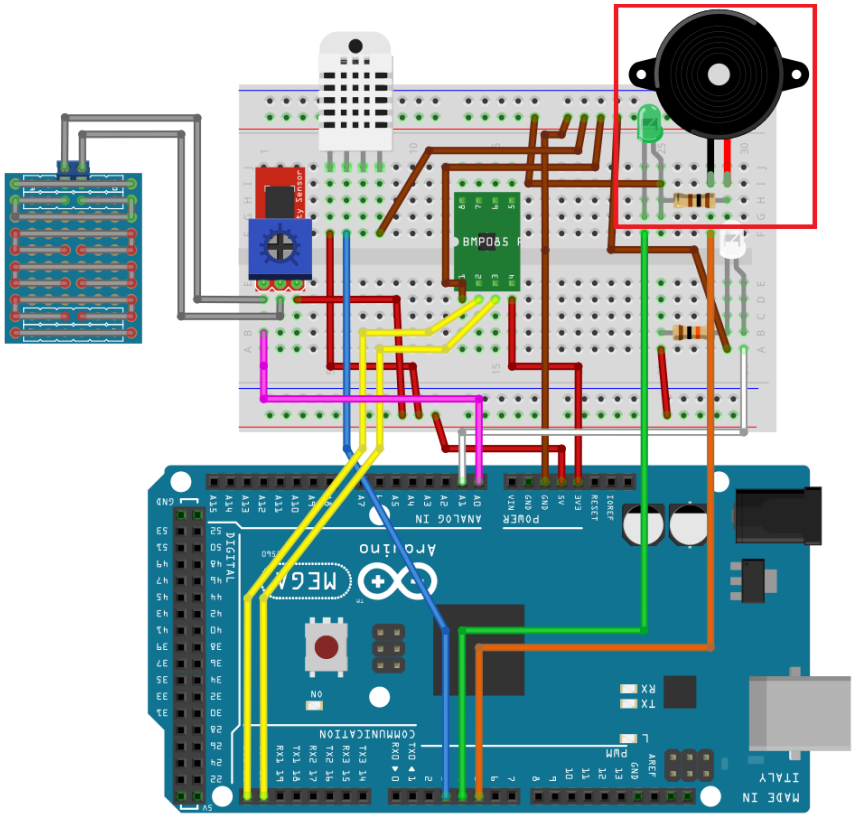
3.5 ALARMA SONORA I VISUAL

Finalment per acabar amb el muntatge de components a la nostra placa Arduino instal·larem un brunzidor Buzzer passiu que genera un xiulet agut, per crear la alarma sonora, i un díode LED verd per crear la alarma visual.

La idea consisteix a sincronitzar el parpelleig del LED amb el xiulet emès per el Buzzer (similar a les alarmes d'un vehicle), de manera que notifiqui d'un possible estat d'alerta com a canvis meteorològics, a l'inici d'una pluja o tempesta.

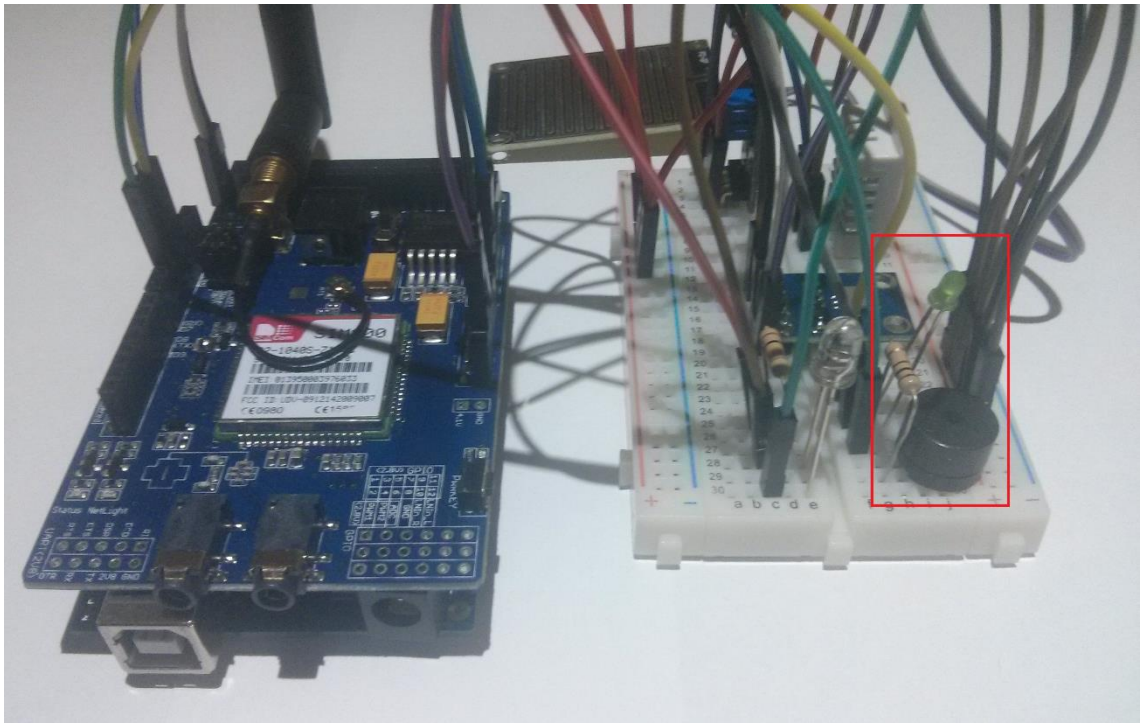
A continuació tractarem de simular aquest sistema, que integrarem posteriorment en el programa principal com una funció més, capaç de activar l'alarma.

Adjuntem l'esquema del muntatge corresponent al sistema, com veiem a la següent il·lustració, és recomanable col·locar una resistència de 100Ω entre l'extrem positiu del Buzzer i la massa. L'altra patilla del Buzzer la connectem directament l'entrada digital número 5, i el LED l'entrada digital número 4 de la placa Arduino.



Il·lustració 61: *Esquema connexió sensor Temperatura i Humitat + sensor Pressió Atmosfèrica i Altitud + sensor lluminositat + sensor de pluja + Buzzer i LED*

Procedim a realitzar el muntatge anterior a la placa Arduino més el mòdul GSM i els sensors que havíem afegit anteriorment.



Il·lustració 62: *Muntatge mòdul GSM + sensor Temperatura i Humitat + sensor Pressió Atmosfèrica i Altitud + sensor lluminositat + sensor de pluja + Buzzer i LED*

A continuació passem a generar el codi que activi l'alarma sonora i visual 3 cops amb intervals periòdics de un segon.

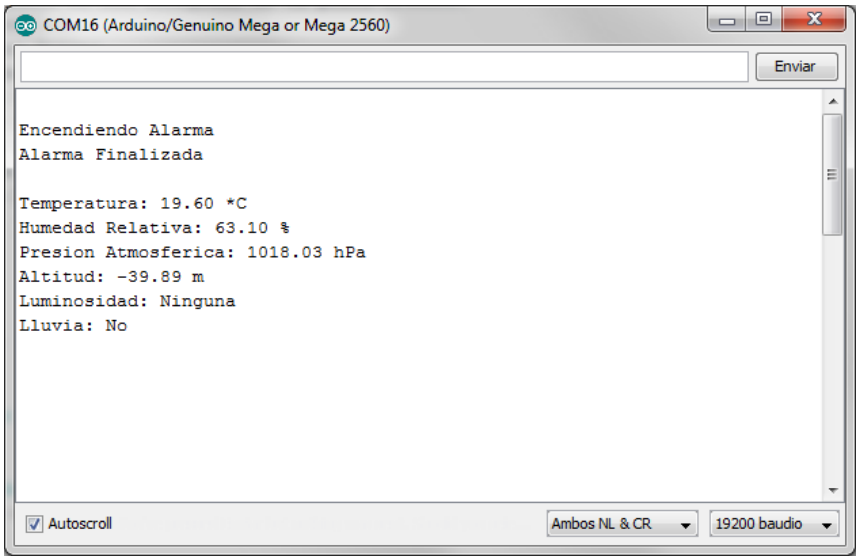
```
#include <Wire.h>
#include <SoftwareSerial.h>
#include <DHT.h>
#include <BMP085.h>

#define _SS_MAX_RX_BUFF 128
#define DHTPIN 3
#define DHTTYPE DHT22

int Led_placa = 13;
int Led = 4;
int Buzzer = 5;
int Alarma = 1;
float t = 0;
float h = 0;
float p = 0;
float a = 0;
int llv = 0;
int lm = 0;
```

```
String Numtelefono1 = "+34666666666";
DHT dht(DHTPIN, DHTTYPE);
BMP085 bmp;
SoftwareSerial SIM900(7,8);

void setup()
{
  Serial.begin(19200);
  dht.begin();
  bmp.begin();
  SIM900.begin(19200);
  pinMode(Led_placa,OUTPUT);
  pinMode(Led,OUTPUT);
  pinMode(Buzzer,OUTPUT);
  t = dht.readTemperature();
  h = dht.readHumidity();
  p = bmp.readPressure();
  a = bmp.readAltitude();
  llv = analogRead(A0);
  lm = analogRead(A1);
  Encender_Alarma();
}

void loop()
{
  Serial.print("\nTemperatura: ");
  Serial.print(t);
  Serial.print(" *C");
  Serial.print("\nHumedad Relativa: ");
  Serial.print(h);
  Serial.print(" %");
  Serial.print("\nPresion Atmosferica: ");
  Serial.print(p/100);
  Serial.print(" hPa");
  Serial.print("\nAltitud: ");
  Serial.print(a);
  Serial.print(" m");
  if (lm >= 90)
    {Serial.print("\nLuminosidad: Alta");}
    else if (lm >= 80)
    {Serial.print("\nLuminosidad: Media");}
    else if (lm >= 70)
    {Serial.print("\nLuminosidad: Baja");}
    else
    {Serial.print("\nLuminosidad: Ninguna");}
  if (llv <= 270)
    {Serial.print("\nLluvia: Si\n");
    Encender_Alarma();}
    else
    {Serial.print("\nLluvia: No\n");}
  delay(5000);
  t = dht.readTemperature();
  h = dht.readHumidity();
  p = bmp.readPressure();
  a = bmp.readAltitude();
  llv = analogRead(A0);
  lm = analogRead(A1);
}

void Encender_Alarma()
{
```

```
  Serial.print("\nEncendiendo Alarma");
  while (Alarma <= 3)
  {
    digitalWrite(Led,HIGH);
    analogWrite(Buzzer,250);
    delay(1000);
    digitalWrite(Led,LOW);
    analogWrite(Buzzer,LOW);
    delay(1000);
    Alarma++;
  }
  Alarma = 1;
  Serial.print("\nAlarma Finalizada\n");
}
```

Un cop compilat, hem d'assegurar que tenim connectat el nostre sistema al port USB del PC junt amb la font d'alimentació externa, i procedim amb la càrrega sketch a la memòria d'Arduino.

Finalitzat el procés de càrrega, el sistema d'alerta s'activarà immediatament, fent parpellejar el LED i emetent el xiulet en sincronia a través del Buzzer, tres pics amb intervals periòdics de un segon. Podem observar el missatge amb les dades obtingudes en el Monitor Serial a la següent il·lustració.



Il·lustració 63: *Activar l'alarma + dades de Temperatura , Humitat, Pressió Atmosfèrica, Altitud, lluminositat i pluja en el Monitor Serial*

4. DESENROTLLAMENT

En aquest apartat durem a terme el procés final de programació del sketch principal, on inclourem totes les funcions automatitzades anomenades a la introducció del treball, i així haurem completat totes les funcionalitats que tindrà l'estació meteorològica aïllada.

Fins ara amb programat l'Arduino perquè fos capaç de realitzar les següents funcions:

- Enviar SMS.
- Rebre SMS.
- Mesurar la Temperatura.
- Mesurar la Humitat Relativa.
- Mesurar la Pressió Atmosfèrica.
- Mesurar l'Altitud.
- Mesurar la Lluminositat.
- Detectar la si esta plovent.
- Emetre una alarma sonora i visual.

Així doncs tenim que combinar totes aquestes funcions per poder crear l'estació meteorològica aïllada amb els següents automatismes:

- Avis mitjançant un SMS el detectar la pluja.
- Avis mitjançant un SMS el detectar una tempesta.
- Sol·licitar per SMS les dades meteorològiques actuals.
- Equipar tota la plataforma amb un equip de energia autosostenible format per una placa solar i una bateria, per obtenir una alimentació interrompuda les 24h del dia.

4.1 DETECTAR PLUJA

La pluja és un fenomen interessant de la natura. A diferència d'altres tants similars més poderosos i destructius, com ara tornats o huracans, aquesta pot ser observada plàcidament i fins i tot gaudida depenent de les condicions, tot i que també pot causar un desastre natural, com les inundacions.

La pluja és un fenomen atmosfèric que s'inicia amb la condensació del vapor d'aigua que contenen els núvols. El seu origen es deu als canvis de pressió, temperatura a l'atmosfera i per la disponibilitat d'aigua en el medi. En concret la pluja depèn de tres factors: la pressió, la temperatura i especialment de la radiació solar.

L'atmosfera sempre té un percentatge d'aigua determinat en forma de vapor, com més elevada sigui la temperatura a l'atmosfera, més capacitat tendra de evaporar-se.
Aquesta aigua de pluja pot condensar-se i a continuació precipitar-se per distintes causes[18].

Il·lustració 64: *Cicle de l'aigua*

Si entra en contacte amb un front fred, l'atmosfera es refreda i serà menys capaç de transportar vapor d'aigua, aquesta es condensa i plou, ja que el fred baixa el grau de saturació col·licionant amb un obstacle natural.

Les petites partícules de pols suspeses en l'atmosfera també fan la funció de nuclis de concentració. A Espanya a vegades plou aigua bruta de terra que prové del desert del Sàhara i l'aigua es condensa quan entra en contacte amb ella.

En les últimes dècades s'ha produït un fenomen que causa pluges amb major freqüència a la nit quan la radiació solar és menor.

Existeixen també altres fenòmens curiosos vinculats a la pluja, com pot ser per exemple la pluja àcida, bastant perjudicial per al medi ambient, o també d'altres més agradables com l'arc de Sant Martí.

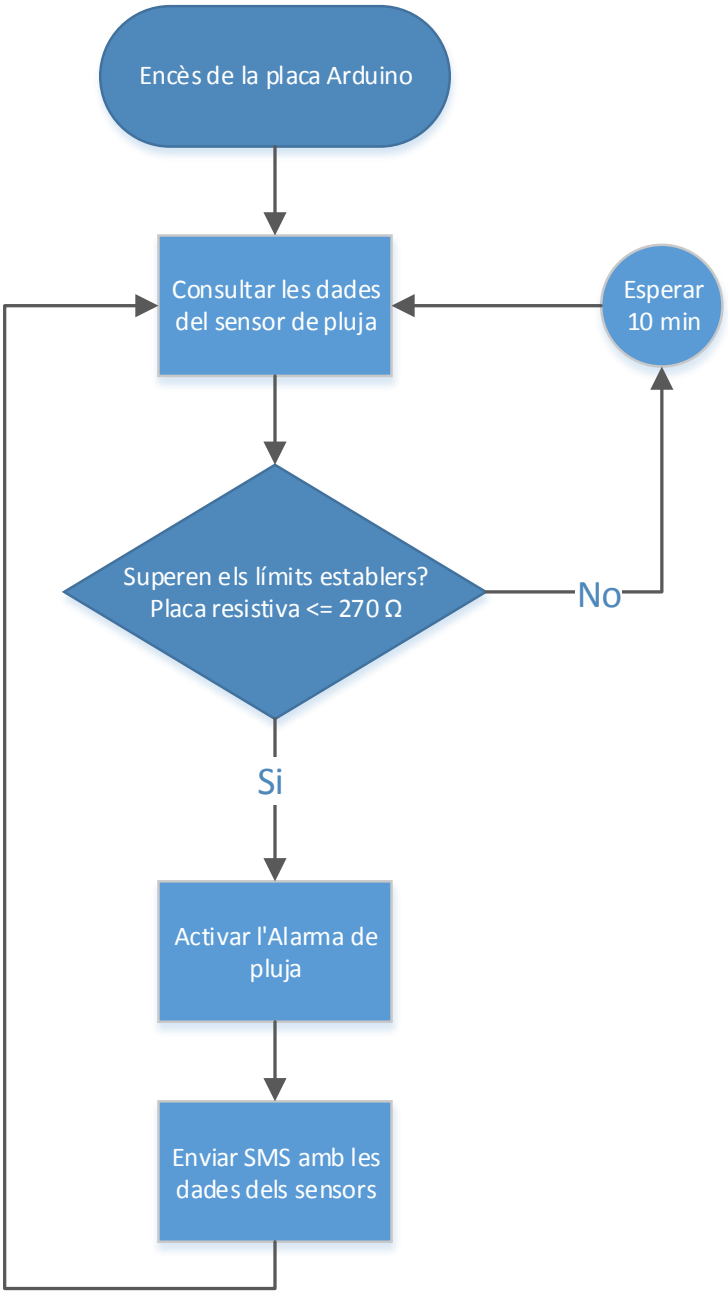
Per detectar una pluja haurem de seguint el següent criteri:

- La placa resistiva que detecta les gotes d'aigua obtingui un valor de 270 Ω o menor.

Si es compleixen aquets criteri s'enviarà un SMS el destinatari informant del esdeveniment junt amb les dades meteorològiques actuals, i s'activarà la alarma sonora i visual.

Aquets criteri és aproximat, la qual cosa s'obtindran les mesures en les pròximes pluges per ajustar el paràmetre del sensor meteorològic el màxim possible.

A continuació es presenta el diagrama de flux per detectar pluges:



A continuació procedim a generar el codi necessari per detectar la pluja i activar la alerta.

```
#include <Wire.h>
#include <DHT.h>
#include <BMP085.h>

#define DHTPIN 3
#define DHTTYPE DHT22

int LED_placa = 13;
int LED = 4;
int Buzzer = 5;
int Alarma = 1;
int contarBuffer=0;
int Tiempo_Tormenta = 4320;
int Tiempo_Lluvia = 4320;
unsigned char buffer[64];
float t = 0;
float h = 0;
float p = 0;
float a = 0;
int llv = 0;
int lm = 0;
boolean Aviso_Tormenta = false;
boolean Aviso_Lluvia = false;
String Numtelefono1 = "+34666666666";
DHT dht(DHTPIN, DHTTYPE);
BMP085 bmp;

void setup()
{
  Serial.begin(19200);
  Serial1.begin(19200);
  dht.begin();
  bmp.begin();
  pinMode(LED_placa,OUTPUT);
  pinMode(LED,OUTPUT);
  pinMode(Buzzer,OUTPUT);
  t = dht.readTemperature();
  h = dht.readHumidity();
  p = bmp.readPressure();
  a = bmp.readAltitude();
  llv = analogRead(A0);
  lm = analogRead(A1);
  Encender_GSM();
  Aviso_Lluvia = true;
  SMS_Datos();
}

void loop()
{
  Tiempo_Tormenta++;
  Tiempo_Lluvia++;
  t = dht.readTemperature();
  h = dht.readHumidity();
  p = bmp.readPressure();
  a = bmp.readAltitude();
  llv = analogRead(A0);
  lm = analogRead(A1);
```

```
  Serial.print("\nTemperatura: ");
  Serial.print(t);
  Serial.print(" *C");
  Serial.print("\nHumedad Relativa: ");
  Serial.print(h);
  Serial.print(" %");
  Serial.print("\nPresion Atmosferica: ");
  Serial.print(p/100);
  Serial.print(" hPa");
  Serial.print("\nAltitud: ");
  Serial.print(-a);
  Serial.print(" m");
  if (lm >= 90)
    {Serial.print("\nLuminosidad: Alta");}
    else if (lm >= 80)
    {Serial.print("\nLuminosidad: Media");}
    else if (lm >= 70)
    {Serial.print("\nLuminosidad: Baja");}
    else
    {Serial.print("\nLuminosidad: Ninguna");}
  if (llv <= 270)
  {
    Serial.print("\nLluvia: Si\n");
    Aviso_Lluvia = true;
    SMS_Datos();
  }
  else
  {Serial.print("\nLluvia: No\n");}
  if (t < 18 && h > 80 && p < 970 && lm <= 80)
  {
    Aviso_Tormenta = true;
    SMS_Datos();
  }
  delay(5000);
}

void SMS_Datos()
{
  Serial.println("\nEnviando SMS");
  Serial1.print("AT\r");
  delay(1000);
  Serial1.print("AT+CMGF=1\r");
  delay(1000);
  Serial1.println("AT+CMGS=\""+Numtelefono1+"\"");
  delay(1000);
  if (Aviso_Lluvia == true && Tiempo_Lluvia >= 4320)
  {
    Serial.print("Aviso de Lluvia");
    Serial1.print("Aviso de Lluvia!!\n\n");
    Encender_Alarma();
    Aviso_Lluvia = false;
    Tiempo_Lluvia = 0;
  }
  if (Aviso_Tormenta == true && Tiempo_Tormenta >= 4320)
  {
    Serial.print("Aviso de Tormenta");
    Serial1.print("Aviso de Tormenta!!\n\n");
    Encender_Alarma();
    Aviso_Tormenta = false;
    Tiempo_Tormenta = 0;
  }
```

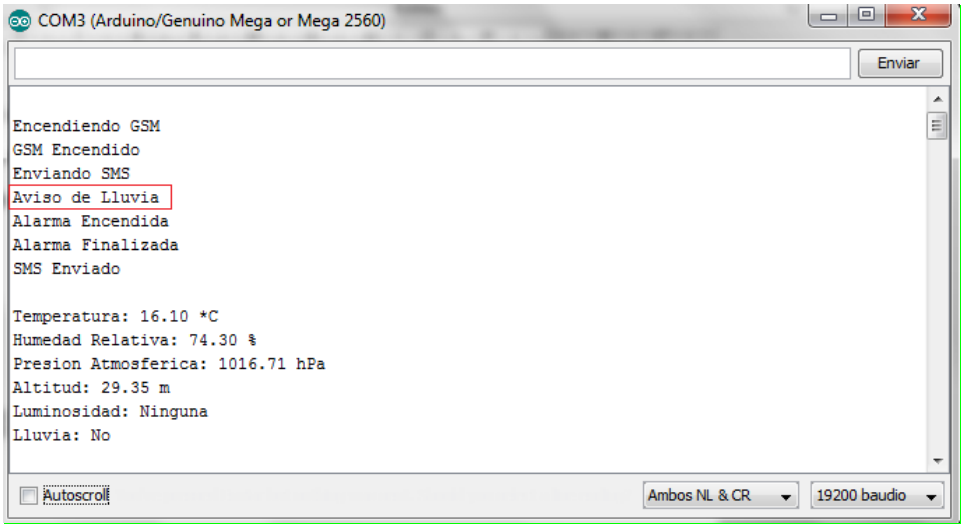
```
  Serial1.print("Temperatura: ");
  Serial1.print(t);
  Serial1.print(" *C");
  Serial1.print("\nHumedad Relativa: ");
  Serial1.print(h);
  Serial1.print(" %");
  Serial1.print("\nPresion Atmosferica: ");
  Serial1.print(p/100);
  Serial1.print(" hPa");
  Serial1.print("\nAltitud: ");
  Serial1.print(-a);
  Serial1.print(" m");
  if (lm >= 90)
    {Serial1.print("\nLuminosidad: Alta");}
    else if (lm >= 80)
    {Serial1.print("\nLuminosidad: Media");}
    else if (lm >= 70)
    {Serial1.print("\nLuminosidad: Baja");}
    else
    {Serial1.print("\nLuminosidad: Ninguna");}
  if (llv <= 270)
    {Serial1.print("\nLluvia: Si");}
    else
    {Serial1.print("\nLluvia: No");}
  Serial1.write((char)26);
  delay(8000);
  Serial.print("SMS Enviado\n");
}

void Encender_Alarma()
{
  Serial.print("\nAlarma Encendida");
  while (Alarma <= 3)
  {
    digitalWrite(LED,HIGH);
    analogWrite(Buzzer,250);
    delay(1000);
    digitalWrite(LED,LOW);
    analogWrite(Buzzer,LOW);
    delay(1000);
    Alarma++;
  }
  Alarma = 1;
  Serial.print("\nAlarma Finalizada\n");
}

void Encender_GSM()
{
  Serial.print("\nEncendiendo GSM");
  pinMode(9, OUTPUT);
  digitalWrite(9,LOW);
  delay(1000);
  digitalWrite(9,HIGH);
  delay(2000);
  digitalWrite(9,LOW);
  delay(3000);
  delay(30000);
  Serial.print("\nGSM Encendido");
}
```

Un cop compilat, hem d'assegurar-nos que tenim connectat la nostre placa el port USB del PC junt amb la font d'alimentació externa, i a continuació procedim a càrrega el sketch a la memòria de l'Arduino.

Finalitzat el procés de càrrega, forçarem que el sistema d'alerta de pluja s'activi immediatament, fent parpellejar el LED i emetent el xiulet en sincronia a través del Buzzer, tres pics amb intervals periòdics de un segon. Podem observar el SMS rebut en el nostre mòbil, amb les dades obtingudes.

Il·lustració 65: *SMS rebut detectar pluja*

També podem observar el missatge amb les dades obtingudes en el Monitor Serial a la següent il·lustració.



Il·lustració 66: *Detectar pluja en el Monitor Serial*

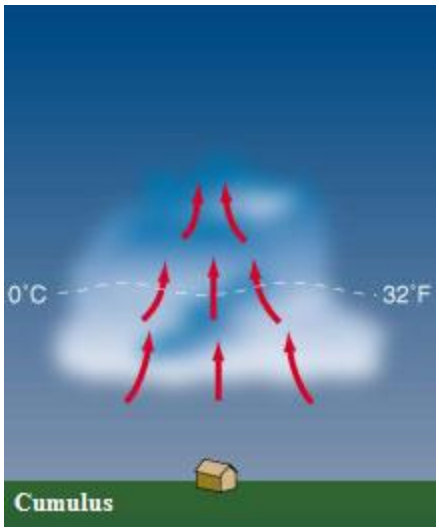
4.2 DETECTAR TEMPESTA

Les tempestes són un dels fenòmens naturals i meteorològics més interessants i sorprenents del nostre planeta.

Per a la formació de tempestes es necessita la coexistència de dues masses d'aire de diferents temperatures. Bàsicament ocorren quan un centre de baixa pressió es genera dins d'un sistema d'alta pressió.
Tota aquesta activitat és capaç de generar una gran variació a l'aire, ja que el contrast tèrmic i la interacció de les masses d'aire humit generen moviments d'aire que es tradueixen en vents i pluges, i fins i tot descàrregues elèctriques quan s'assoleix la tensió de ruptura de l'aire .
Les tempestes es poden generar en diferents magnituds, no és el mateix una tempesta tropical amb vents huracanats, que una simple tempesta que no produeix danys significatius.

Les tempestes es creen en tres etapes:

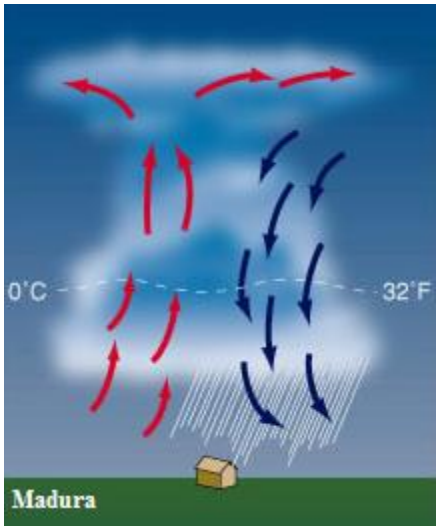
- **Etapa Cumulus:** El sol escalfa la superfície de la Terra durant el dia. La calor de la superfície escalfa l'aire proper. Com l'aire calent és més lleuger que l'aire fresc, comença a elevar-se (conegut com a corrent ascendent). Si l'aire és humit, llavors l'aire calent es condensa en un núvol cumulus. El núvol continuarà creixent mentre hi hagi aire càlid ascendint.



Il·lustració 67: *Formació etapa Cumulus*

- **Etapa Madura:** Quan el núvol cumulus es fa molt gran, l'aigua en aquest es fa molt pesat. Gotes de pluja comencen a caure pel núvol quan l'aire ascendent ja no pot sostenir-les. Mentrestant, l'aire fred comença a entrar en el núvol. Com l'aire fred és més pesat que l'aire calent, comença a descendir dintre del núvol (conegut com a corrent descendent). El corrent descendent arrossega la pesada aigua cap avall, provocant pluja.

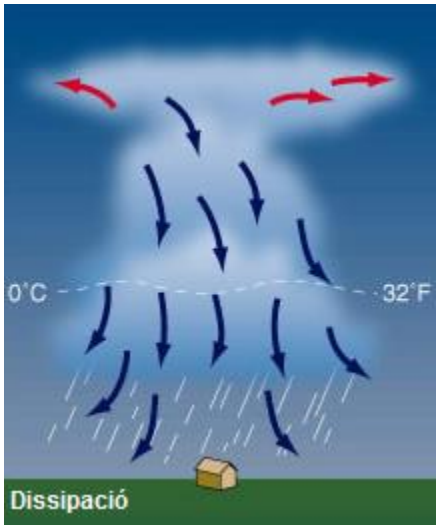
  Aquest núvol s'ha convertit en un núvol cumulonimbus perquè té un corrent ascendent, un corrent descendent, i pluja. Al costat de la forta pluja començaran a produir-se trons i llamps. El núvol cumulonimbus és ara una cel·la de tempesta.



Il·lustració 68: *Formació etapa Madura*

- **Etapa de Dissipació:** Després d'uns 30 minuts, la tempesta comença a dissipar-se. Això es produeix quan el corrent descendent comença a dominar sobre l'ascendent. Com l'aire calent ja no pot elevar-se, no es poden formar més gotes d'aigua. La tempesta desapareix amb una pluja feble mentre els núvols desapareixen de baix cap a dalt.

  El procés complet demora prop d'una hora per tempestes ordinàries. Tempestes supercel·les són molt més grans i més poderoses, i duren diverses hores.



Il·lustració 69: *Formació etapa Dissipació*

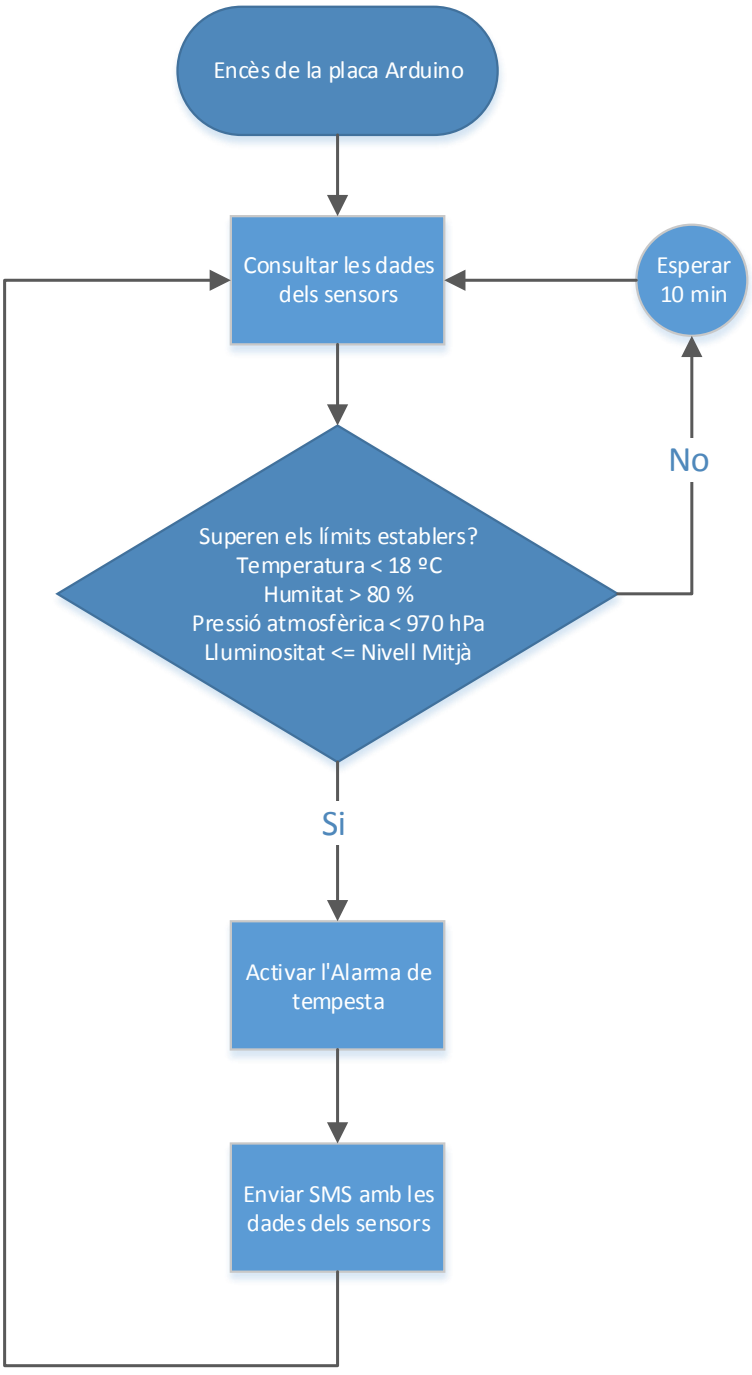
Per detectar una tempesta haurem de seguint el següents criteris:

- Temperatura < 18 ºC.
- Humitat > 80 %.
- Pressió atmosfèrica < 970 hPa.
- Lluminositat <= Nivell Mitjà.

Si es compleixen aquets 4 criteris s'enviarà un SMS el destinatari informant del esdeveniment junt amb les dades meteorològiques actuals, a continuació s'activarà la alarma sonora i visual.

Aquets criteris son aproximats, la qual cosa s'obtindran les mesures en les pròximes tempestes per ajustar els paràmetres dels sensors meteorològics el màxim possible.

A continuació es presenta el diagrama de flux per detectar tempestes:



A continuació procedim a generar el codi necessari per detectar una tempesta i activar la alerta.

```
#include <Wire.h>
#include <DHT.h>
#include <BMP085.h>

#define DHTPIN 3
#define DHTTYPE DHT22

int LED_placa = 13;
int LED = 4;
int Buzzer = 5;
int Alarma = 1;
int contarBuffer=0;
int Tiempo_Tormenta = 4320;
unsigned char buffer[64];
float t = 0;
float h = 0;
float p = 0;
float a = 0;
int llv = 0;
int lm = 0;
boolean Aviso_Tormenta = false;
String Numtelefono1 = "+34666666666";
DHT dht(DHTPIN, DHTTYPE);
BMP085 bmp;

void setup()
{
  Serial.begin(19200);
  Serial1.begin(19200);
  dht.begin();
  bmp.begin();
  pinMode(LED_placa,OUTPUT);
  pinMode(LED,OUTPUT);
  pinMode(Buzzer,OUTPUT);
  t = dht.readTemperature();
  h = dht.readHumidity();
  p = bmp.readPressure();
  a = bmp.readAltitude();
  llv = analogRead(A0);
  lm = analogRead(A1);
  Encender_GSM();
  Aviso_Tormenta = true;
  SMS_Datos();
}

void loop()
{
  Tiempo_Tormenta++;
  t = dht.readTemperature();
  h = dht.readHumidity();
  p = bmp.readPressure();
  a = bmp.readAltitude();
  llv = analogRead(A0);
  lm = analogRead(A1);
  Serial.print("\nTemperatura: ");
  Serial.print(t);
```

```
  Serial.print(" *C");
  Serial.print("\nHumedad Relativa: ");
  Serial.print(h);
  Serial.print(" %");
  Serial.print("\nPresion Atmosferica: ");
  Serial.print(p/100);
  Serial.print(" hPa");
  Serial.print("\nAltitud: ");
  Serial.print(-a);
  Serial.print(" m");
  if (lm >= 90)
    {Serial.print("\nLuminosidad: Alta");}
    else if (lm >= 80)
    {Serial.print("\nLuminosidad: Media");}
    else if (lm >= 70)
    {Serial.print("\nLuminosidad: Baja");}
    else
    {Serial.print("\nLuminosidad: Ninguna");}
  if (llv <= 270)
  {
    Serial.print("\nLluvia: Si\n");
  }
  else
  {Serial.print("\nLluvia: No\n");}
  if (t < 18 && h > 80 && p < 970 && lm <= 80)
  {
    Aviso_Tormenta = true;
    SMS_Datos();
  }
  delay(5000);
}

void SMS_Datos()
{
  Serial.println("\nEnviando SMS");
  Serial1.print("AT\r");
  delay(1000);
  Serial1.print("AT+CMGF=1\r");
  delay(1000);
  Serial1.println("AT+CMGS=\""+Numtelefono1+"\"");
  delay(1000);
  if (Aviso_Tormenta == true && Tiempo_Tormenta >= 4320)
  {
    Serial.print("Aviso de Tormenta");
    Serial1.print("Aviso de Tormenta!!\n\n");
    Encender_Alarma();
    Aviso_Tormenta = false;
    Tiempo_Tormenta = 0;
  }
  Serial1.print("Temperatura: ");
  Serial1.print(t);
  Serial1.print(" *C");
  Serial1.print("\nHumedad Relativa: ");
  Serial1.print(h);
  Serial1.print(" %");
  Serial1.print("\nPresion Atmosferica: ");
  Serial1.print(p/100);
  Serial1.print(" hPa");
  Serial1.print("\nAltitud: ");
  Serial1.print(-a);
  Serial1.print(" m");
```

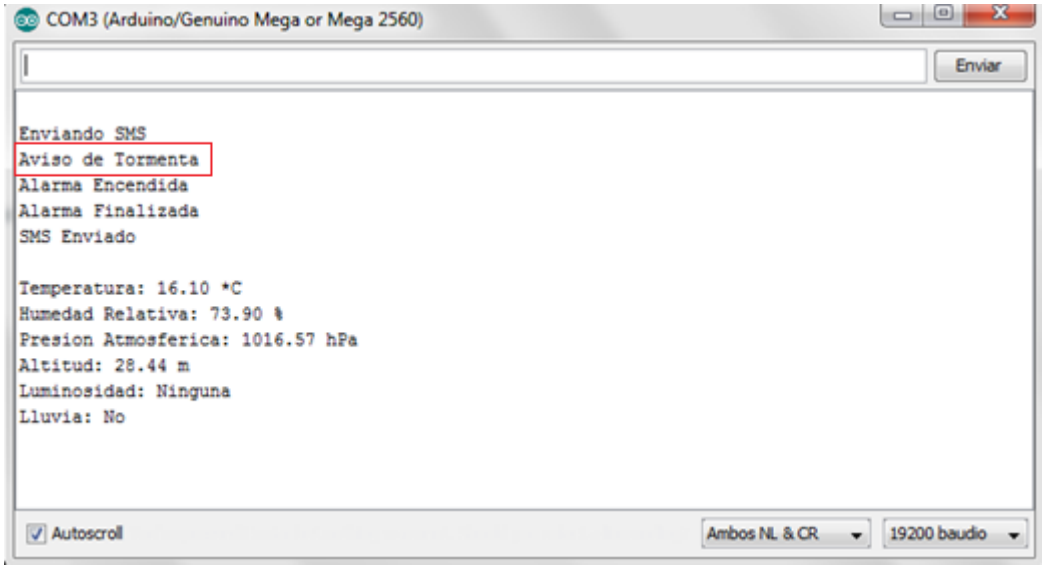
```
  if (lm >= 90)
    {Serial1.print("\nLuminosidad: Alta");}
    else if (lm >= 80)
    {Serial1.print("\nLuminosidad: Media");}
    else if (lm >= 70)
    {Serial1.print("\nLuminosidad: Baja");}
    else
    {Serial1.print("\nLuminosidad: Ninguna");}
  if (llv <= 270)
    {Serial1.print("\nLluvia: Si");}
    else
    {Serial1.print("\nLluvia: No");}
  Serial1.write((char)26);
  delay(8000);
  Serial.print("SMS Enviado\n");
}

void Encender_Alarma()
{
  Serial.print("\nAlarma Encendida");
  while (Alarma <= 3)
  {
    digitalWrite(LED,HIGH);
    analogWrite(Buzzer,250);
    delay(1000);
    digitalWrite(LED,LOW);
    analogWrite(Buzzer,LOW);
    delay(1000);
    Alarma++;
  }
  Alarma = 1;
  Serial.print("\nAlarma Finalizada\n");
}

void Encender_GSM()
{
  Serial.print("\nEncendiendo GSM");
  pinMode(9, OUTPUT);
  digitalWrite(9,LOW);
  delay(1000);
  digitalWrite(9,HIGH);
  delay(2000);
  digitalWrite(9,LOW);
  delay(3000);
  delay(30000);
  Serial.print("\nGSM Encendido");
}
```

Un cop compilat, hem d'assegurar-nos que tenim connectat la nostre placa el port USB del PC junt amb la font d'alimentació externa, i a continuació procedim a càrrega el sketch a la memòria de l'Arduino.

Finalitzat el procés de càrrega, forçarem que el sistema d'alerta de tempesta s'activi immediatament, fent parpellejar el LED i emetent el xiulet en sincronia a través del Buzzer, tres pics amb intervals periòdics de un segon. Podem observar el SMS rebut en el nostre mòbil, amb les dades obtingudes.

Il·lustració 70: *SMS rebut detectar tempesta*

També podem observar el missatge amb les dades obtingudes en el Monitor Serial a la següent il·lustració.



Il·lustració 71: *Detectar tempesta en el Monitor Serial*

4.3 SOL·LICITAR DADES METEOROLÒGIQUES PER SMS

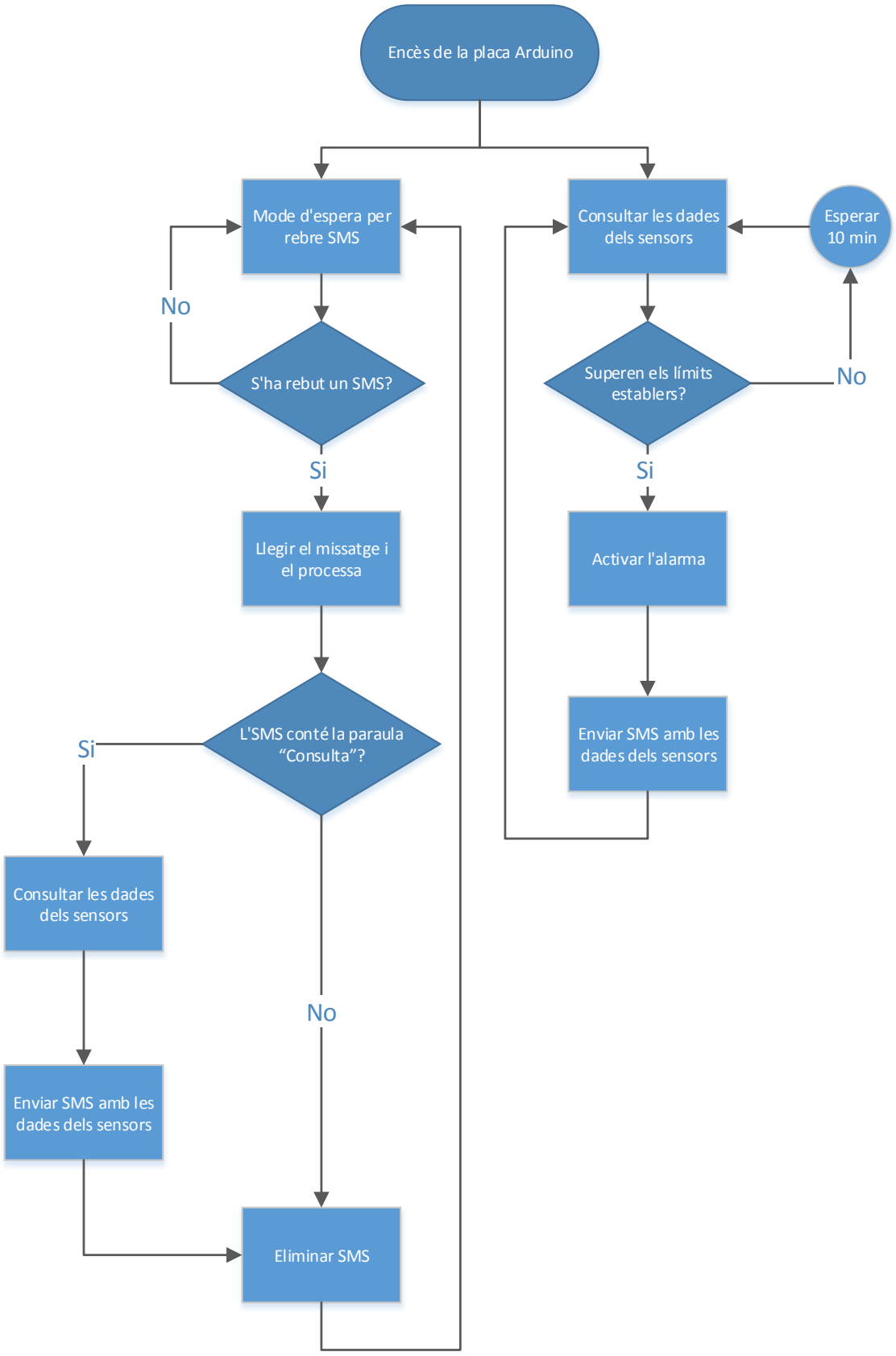
Aquest apartat conté l'algoritme més complicat de tot el programa, ja que s'han implementat les següents funcions:

- Comprovar la memòria SIM per detectar si amb rebut un SMS.
- Llegir el contingut del SMS.
- Cercar en el contingut del SMS la paraula "Consulta."
- Llegir el número de telèfon del qual prové SMS.
- Eliminar SMS.

Aquestes funcions es repetiran contínuament cada 10 minuts, en el moment que l'estació detecti un SMS amb la paraula "Consulta." procedirem a enviar un SMS que contengui les dades actuals dels sensors meteorològics, el mateix número de telèfon que ha realitzat la consulta.

Per finalitzar eliminarem el SMS de la memòria SIM de tal manera que mai estigui plena, i així sempre estarà disponible per rebent SMS nous.

A continuació es presenta el diagrama de flux del sistema de consulta per SMS amb l'estació meteorològica.



A continuació procedim a generar el codi necessari per detectar les consultes per SMS.

```
#include <Wire.h>
#include <DHT.h>
#include <BMP085.h>

#define _SS_MAX_RX_BUFF 128
#define DHTPIN 3
#define DHTTYPE DHT22

int LED = 4;
int Buzzer = 5;
int Alarma = 1;
int num = 0;
int num_borrar = 0;
int contarBuffer = 0;
int Tiempo_Tormenta = 4320;
int Tiempo_Lluvia = 4320;
char telf[11];
unsigned char buffer[200];
float t = 0;
float h = 0;
float p = 0;
float a = 0;
boolean Aviso_Tormenta = false;
boolean Aviso_Lluvia = false;
boolean consulta = false;
boolean eliminar_SMS = false;
String Numtelefono;
String Numtelefono1 = "+34666666666";
DHT dht(DHTPIN, DHTTYPE);
BMP085 bmp;

void setup()
{
  Serial.begin(19200);
  Serial1.begin(19200);
  dht.begin();
  bmp.begin();
  pinMode(LED,OUTPUT);
  pinMode(Buzzer,OUTPUT);
  pinMode(13,OUTPUT);
  Encender_GSM();
}

void loop()
{
  Tiempo_Tormenta++;
  Tiempo_Lluvia++;
  t = dht.readTemperature();
  h = dht.readHumidity();
  p = bmp.readPressure();
  a = bmp.readAltitude();
  Serial.print("\nTemperatura: ");
  Serial.print(t);
  Serial.print(" *C");
  Serial.print("\nHumedad Relativa: ");
  Serial.print(h);
  Serial.print(" %");
```

```
  Serial.print("\nPresion Atmosferica: ");
  Serial.print(p/100);
  Serial.print(" hPa");
  Serial.print("\nAltitud: ");
  Serial.print(a);
  Serial.print(" m");
  if (analogRead(A1) >= 90)
    {Serial.print("\nLuminosidad: Alta");}
    else if (analogRead(A1) >= 70)
    {Serial.print("\nLuminosidad: Media");}
    else if (analogRead(A1) >= 50)
    {Serial.print("\nLuminosidad: Baja");}
    else
    {Serial.print("\nLuminosidad: Ninguna");}
  if (analogRead(A0) <= 270)
  {
    Serial.print("\nLluvia: Si\n");
    Aviso_Lluvia = true;
    Numtelefono = Numtelefono1;
    SMS_Datos();
  }
  else
  {Serial.print("\nLluvia: No\n");}
  if (t < 18 && h > 80 && p < 970 && analogRead(A1) <= 75)
  {
    Aviso_Tormenta = true;
    Numtelefono = Numtelefono1;
    SMS_Datos();
  }
  Recibiendo_SMS();
  delay(5000);
}

void Recibiendo_SMS()
{
  Serial.print("\nMiramos la memoria SIM:\n");
  Serial1.print("AT+CPMS=\"SM\"\r");
  delay(1000);
  Comprobar_Memroia_SIM();
  if (consulta == true)
  {
    Serial1.print("AT+CMGR=1\r");
    delay(135);
    Comprobar_SMS();
    consulta = false;
  }
  if (eliminar_SMS == true)
  {
    Borrar_SMS();
    eliminar_SMS = false;
  }
}

void SMS_Datos()
{
  Serial.print("\nEnviando SMS a: ");
  Serial.print(Numtelefono);
  Serial1.print("AT\r");
  delay(1000);
  Serial1.print("AT+CMGF=1\r");
  delay(1000);
```

```
  Serial1.println("AT+CMGS=\""+Numtelefono+"\"");
  delay(1000);
  if (Aviso_Lluvia == true && Tiempo_Lluvia >= 4320)
  {
    Serial.print("Aviso de Lluvia");
    Serial1.print("Aviso de Lluvia!!\n\n");
    Encender_Alarma();
    Aviso_Lluvia = false;
    Tiempo_Lluvia = 0;
  }
  if (Aviso_Tormenta == true && Tiempo_Tormenta >= 4320)
  {
    Serial.print("Aviso de Tormenta");
    Serial1.print("Aviso de Tormenta!!\n\n");
    Encender_Alarma();
    Aviso_Tormenta = false;
    Tiempo_Tormenta = 0;
  }
  Serial1.print("Temperatura: ");
  Serial1.print(t);
  Serial1.print(" *C");
  Serial1.print("\nHumedad Relativa: ");
  Serial1.print(h);
  Serial1.print(" %");
  Serial1.print("\nPresion Atmosferica: ");
  Serial1.print(p/100);
  Serial1.print(" hPa");
  Serial1.print("\nAltitud: ");
  Serial1.print(a);
  Serial1.print(" m");
  if (analogRead(A1) >= 90)
    {Serial1.print("\nLuminosidad: Alta");}
    else if (analogRead(A1) >= 70)
    {Serial1.print("\nLuminosidad: Media");}
    else if (analogRead(A1) >= 50)
    {Serial1.print("\nLuminosidad: Baja");}
    else
    {Serial1.print("\nLuminosidad: Ninguna");}
  if (analogRead(A0) <= 270)
    {Serial1.print("\nLluvia: Si");}
    else
    {Serial1.print("\nLluvia: No");}
  Serial1.write((char)26);
  delay(8000);
  Serial.println(" SMS Enviado");
}

void Comprobar_Memroia_SIM()
{
  for (int j=0; j<2 ;j++)
  {
    if (Serial1.available())
    {
      while(Serial1.available())
      {
        buffer[contarBuffer++]=Serial1.read();
        if(contarBuffer == 200)break;
      }
      Serial.write(buffer,contarBuffer);
      delay(1000);
      int memoria=0;
```

```
      while (buffer[memoria] <= 200)
      {
      if (buffer[memoria] == 49)
      {memoria++;if (buffer[memoria] == 44)
      {memoria++;if (buffer[memoria] == 53)
      {Serial.println("1 SMS\n"); consulta = true;eliminar_SMS =
true;break;
      }else{memoria = memoria-2;}
      }else{memoria = memoria-1;}
      }memoria++;
      }
      if (consulta == false)
      {Serial.print("0 SMS\n");}
      BorrarBufferArray();
      contarBuffer = 0;
    }
  }
}

void Comprobar_SMS()
{
  num_borrar++;
  for (int j=0; j<2 ;j++)
  {
    if (Serial1.available())
    {
      while(Serial1.available())
      {
        buffer[contarBuffer++]=Serial1.read();
        if(contarBuffer == 200)break;
      }
      Serial.write(buffer,contarBuffer);
      delay(1000);
      if (j==0)
      {
        if (consulta == true)
        {
          int buscar=0;
          while (buffer[buscar] <= 200)
          {
            if (buffer[buscar] == 34)
            {
              buscar++;
              if (buffer[buscar] == 43)
              {
                for (int x=0; x<12 ;x++)
                {
                  telf[x] = buffer[buscar];
                  buscar++;
                }
                Numtelefono = telf;
              }
              else
              {
                buscar = buscar-1;
              }
            }
            buscar++;
          }
        }
      }
```

```
      if (j==1)
      {
        if (consulta == true)
        {
          int buscar=0;
          while (buffer[buscar] <= 200)
          {
          if (buffer[buscar] == 67)
          {buscar++;if (buffer[buscar] == 111)
          {buscar++;if (buffer[buscar] == 110)
          {buscar++;if (buffer[buscar] == 115)
          {buscar++;if (buffer[buscar] == 117)
          {buscar++;if (buffer[buscar] == 108)
          {buscar++;if (buffer[buscar] == 116)
          {buscar++;if (buffer[buscar] == 97)
          {buscar++;if (buffer[buscar] == 46)
          {Serial.println("Palabra
encontrada!");Encender_Alarma_SMS();SMS_Datos();
          }else{buscar = buscar-8;}
          }else{buscar = buscar-7;}
          }else{buscar = buscar-6;}
          }else{buscar = buscar-5;}
          }else{buscar = buscar-4;}
          }else{buscar = buscar-3;}
          }else{buscar = buscar-2;}
          }else{buscar = buscar-1;}
          }buscar++;
          }
        }
      }
      BorrarBufferArray();
      contarBuffer = 0;
    }
  }
}

void BorrarBufferArray()
{
  for (int i=0; i<contarBuffer;i++)
  {
    buffer[i]=NULL;
  }
}

void Encender_Alarma()
{
  Serial.print("\nAlarma Encendida");
  while (Alarma <= 3)
  {
    digitalWrite(LED,HIGH);
    analogWrite(Buzzer,250);
    delay(1000);
    digitalWrite(LED,LOW);
    analogWrite(Buzzer,LOW);
    delay(1000);
    Alarma++;
  }
  Alarma = 1;
  Serial.println(" Alarma Finalizada");
}
```

```
void Encender_Alarma_SMS()
{
  Serial.print("\nAlarma Encendida");
  while (Alarma <= 1)
  {
    digitalWrite(LED,HIGH);
    analogWrite(Buzzer,250);
    delay(4000);
    digitalWrite(LED,LOW);
    analogWrite(Buzzer,LOW);
    Alarma++;
  }
  Alarma = 1;
  Serial.println(" Alarma Finalizada");
}

void Encender_GSM()
{
  Serial.print("\nEncendiendo GSM");
  pinMode(9, OUTPUT);
  digitalWrite(9,LOW);
  delay(1000);
  digitalWrite(9,HIGH);
  delay(2000);
  digitalWrite(9,LOW);
  delay(25000);
  Serial.print(" GSM Encendido");
}

void Borrar_SMS()
{
  if (num_borrar >= 2)
  {
    Serial.println("\nBorramos el SMS de la SIM");
    Serial1.print("\nAT+CMGD=1\r");
    delay(1000);
    num_borrar = 0;
  }
}
```

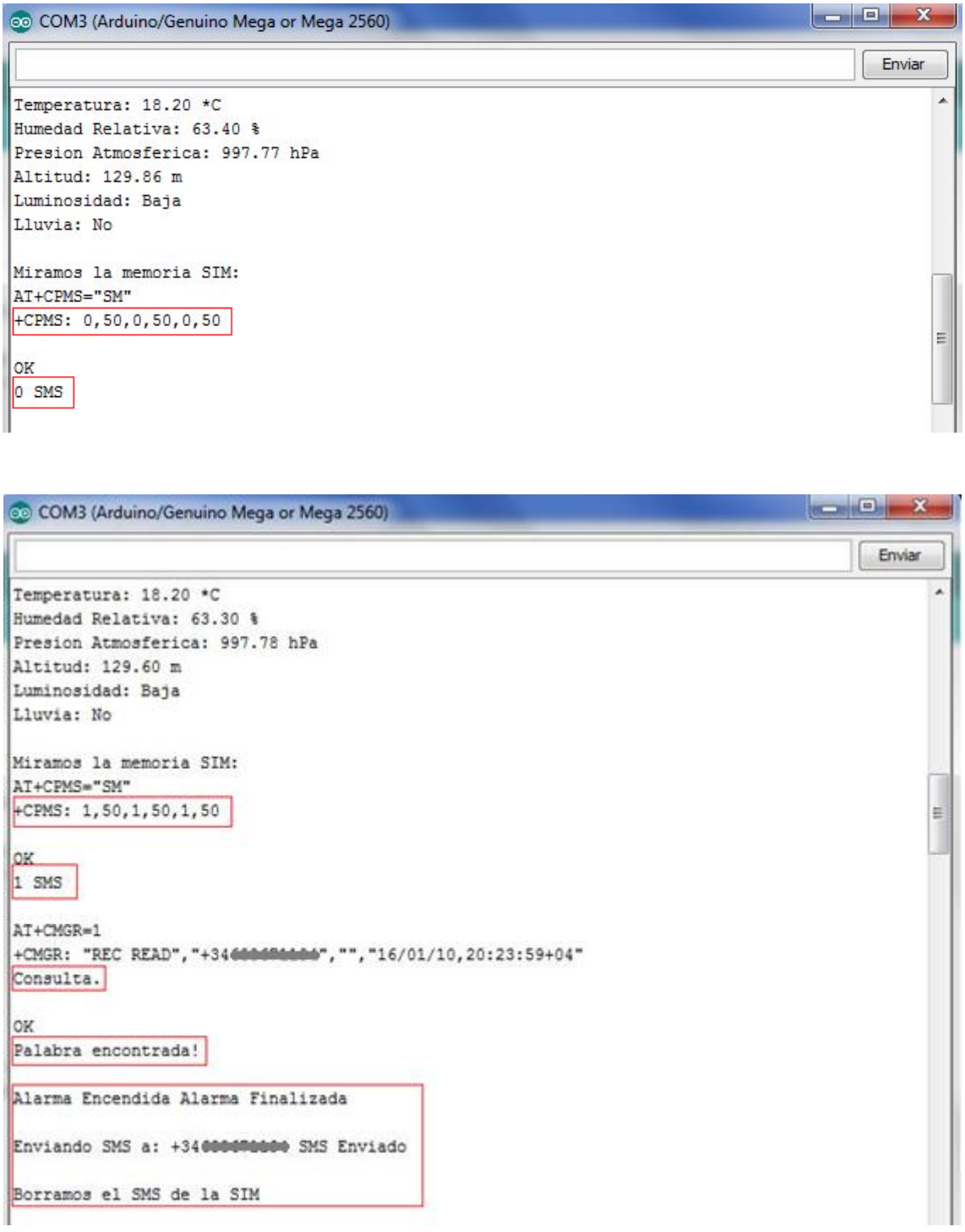
Un cop compilat, hem d'assegurar-nos que tenim connectat la nostre placa el port USB del PC junt amb la font d'alimentació externa, i a continuació procedim a càrrega el sketch a la memòria de l'Arduino.

Finalitzat el procés de càrrega, l'estació començarà a consultar la memòria SIM cada 10 minuts fins que haguí llegit un SMS amb la paraula "Consulta.", a continuació farà parpellejar el LED i emetent el xiulet en sincronia a través del Buzzer, tres pics amb intervals periòdics de un segon, i enviarà les dades meteorològiques actuals el número de telèfon que ha realitzat la consulta.

Podem observar l'SMS rebut en el nostre mòbil, amb les dades obtingudes:

Il·lustració 72: *SMS rebut al realitzar la consulta*

També podem observar el missatge amb les dades obtingudes en el Monitor Serial a la següent il·lustració.



Il·lustració 73: *Detectar SMS rebut en el Monitor Serial*

4.4 SISTEMA AUTÒNOM

Finalment mesurarem el consum energètic que te l'estació meteorològica i d'aquesta manera podrem realitzar els càlculs necessaris per determinar quina bateria i placa solar seran necessàries per mantenir l'estació encesa les 24 hores.

Primer mesura'm el consum d'amperatge de cada component de l'estació:

	Voltatge	**Amperatge**
Arduino MEGA	5 v	41.2 mA
Mòdul GSM	5 v	114.3 mA
Sensor de temperatura i humitat	5 v	2.4 mA
Sensor de pressió atmosfèrica i altitud	3.3 v	0.6 mA
Sensor de lluminositat	5 v	50.1 mA
Sensor de pluja	5 v	7.8 mA
Total	5 v	216.4 mA

Taula 6: *Consum energètic de l'estació meteorològica*

Tenim un consum de 216.4 mA quant la estació esta en funcionament però pot obtenir un consum puntual pic de 523.8 mA quant s'envia un SMS i s'encén l'alarma formada per el díode LED i el Buzzer.

BATERIA

L'objectiu inicial és proporcionar una bateria capaç de mantenir amb funcionament l'estació meteorològica al menys 24 hores sense la ajuda de la placa solar ni ninguna alimentació externa.

$$216.4\ mA/h \ x \ 24h = 5193.6\ mA/h$$

Així docs necessitaríem una bateria de com a mínim 5193.6 mA/h.

En aquest cas amb seleccionarem una bateria Li-Po de 6600 mA/h que ens oferirà una autonomia aproximada de 30.5 hores.

Il·lustració 74: *Bateria 6600 mA/h*

Característiques:

- Voltatge: 5 V
- Capacitat: 6600 mA/h
- Descàrrega màxima: 5 A
- Tecnologia: Li-Po amb circuit de control
- Mesura: 51x65x28 mm
- Pes: 145 g

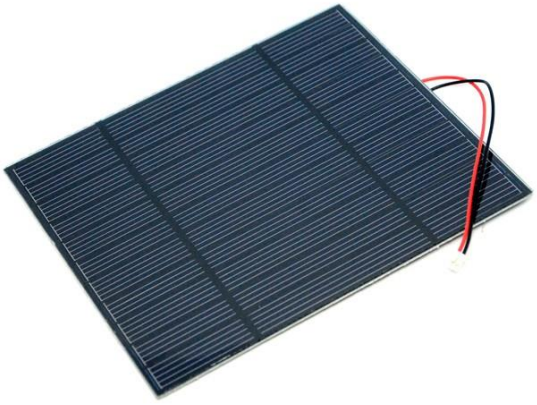
PLACA SOLAR

L'objectiu de la placa solar és alimentar l'estació meteorològica i carregar la bateria en les hores que es produeix radiació solar.
La placa solar a de ser capaç de realitzar una carga completa de la bateria anteriorment seleccionada amb un màxim de 12 hores, que es el temps aproximat de radiació solar en un dia.

6600 mA/h / 12 hores de sol = 550 mA/h

Així docs necessitaríem una placa solar capaç de generar com a mínim 550 mA/h

En aquest cas amb seleccionarem una placa solar de 600 mA/h que ens permetrà carregar la bateria aproximadament amb 11 hores.



Il·lustració 75: *Placa Solar 3.5 W*

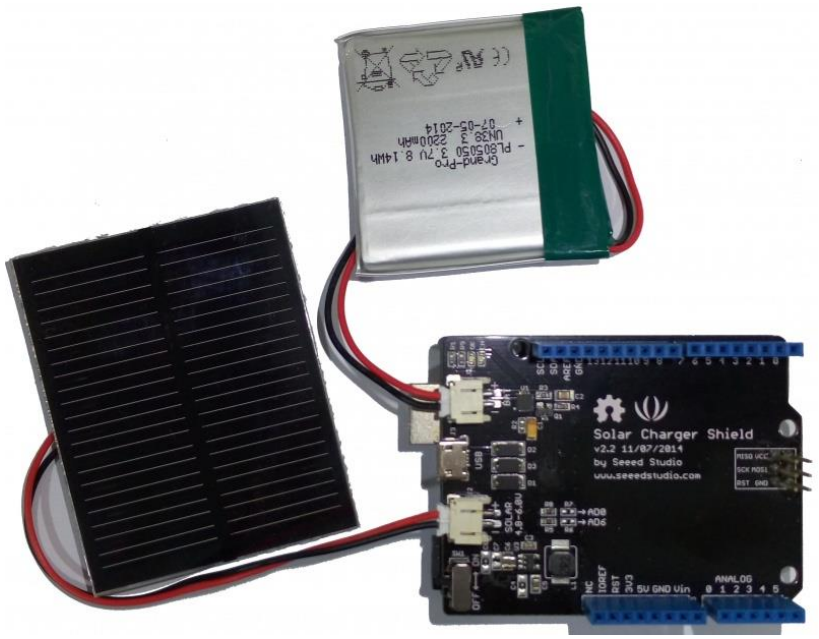
Característiques:

- Voltatge: 5.5 V
- Corrent: 640 mA/h
- Potència: 3.5 W
- Eficiència: 16 %
- Materials: PCB i Fibra de vidre
- Mesura: 138x160x1.5 mm
- Pes: 315 g

MÒDUL DE CARREGA SOLAR

Per finalitzar necessitarem un últim mòdul capaç de gestionar la energia de l'estació meteorològica de forma automàtica.

Aquest tipus de mòdul pot alimentar la nostra estació gestionant tres voltatges d'entrada: la placa solar, la bateria i una entrada externa que ens permetrà carregar la bateria de forma directe mitjançant un cable USB sense dependre de la placa solar.



Il·lustració 76: Mòdul de Carrega Solar v2.2

Característiques:

- Protecció de curtcircuit
- Indicador d'estat de Bateria (Vermell: Carregant, Verd Carregada)
- Corrent de carrega de bateria fins a 900 mA
- Voltatge d'entrada Placa Solar: 4.8 – 6 V
- Voltatge d'entrada Bateria: 3 – 4.5 V
- Voltatge d'entrada USB: 4.75 – 5.25 V
- Potencia màxima de sortida: 3 W (600 mA - 5V)
- Mesura: 68x52 mm
- Pes: 62 g

5. CONCLUSIONS

El treball ha estat una gran experiència, he après molts conceptes de programació sobre la plataforma Arduino i els seus components, aplicat patrons i construït una arquitectura modular utilitzant conceptes adquirits al llarg del grau.
Un altre dels avantatges que me ha aportat realitzar el treball, ha estat en la millora a l'hora de crear documents d'aquesta envergadura, veure els meus errors i així poder millorar radiacions futures.

La planificació no se ha complit a la perfecció, me he atraçat un poc en els terminis prefixats, ja que algunes de les ultimes funcionalitats a desenvolupar m'ha portat més temps del que em pensava inicialment.

Respecte el objectius se han complit en gran part, ja que finalment la plataforma compleix en tots els requisits i funcionalitats que s'havien descrit. Només me ha faltat fer l'encapsulat final per obtenir tota l'estació meteorològica en un format més compacte.

De cara al futur m'agradaria implementar funcionalitats que m'han anat succeint durant la creació del treball. Com afegir més sensors meteorològics o també augmentar les possibilitats llegint diferents ordres remotes per SMS per executar distintes tasques.

D'altra banda, una de les principals intencions del projecte era construir un dispositiu de baix cost que pogués estar a l'abast pràcticament de qualsevol consumidor.
Si analitzem el pressupost total, hem aconseguit mantenir-nos en els límits teòrics de 200€. A la següent taula podem veure el desglossament del pressupost total de l'estació meteorològica.

	Unitats	PVP
Arduino Mega	1	22.00 €
Mòdul GSM SIM900	1	49.95 €
Sensor Temperatura i Humitat	1	7.00 €
Sensor Pressió atmosfèrica i Altitud	1	13.00 €
Sensor de Lluminositat	1	1.89 €
Sensor de Pluja	1	6.00 €
Placa Solar	1	15.00 €
Bateria Li-Po	1	30.00 €
Mòdul Carrega Solar	1	12.60 €
Carregador 5 V	1	5.95 €
Protoboard Mini	1	3.60 €
LED	1	0.80 €
Buzzer	1	1.85 €
Resistències	2	1.50 €
Cablejat	2	8.30 €
Total	17	179.44 €

Taula 7: *Pressupost de l'estació meteorològica*

6. GLOSSARI

AT: *Attention*

EEPROM: *Electrically Erasable Programmable Read-Only Memory*

GPRS: *General Packet Radio Service*

GSM: *Global System for Mobile communications*

hPa: *Hectopascal*

HR: *Humitat relativa*

I2C: *Inter-Integrated Circuit*

IDE: *Integrated Development environment*

IE: *Information Element*

IEEE: *Institute of Electrical and Electronics Engineers*

IoT: *Internet of Things*

IPC: *Inter-process communication*

ISR: *Interrupt Service Routine*

LED: *Light-emitting diode*

SoC: *System on Chip*

SRAM: *Static Random Access Memory*

USB: *Universal Serial Bus*

7. BIBLIOGRAFIA

Referències:

[1] https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

[2] https://www.adafruit.com/datasheets/BMP085_DataSheet_Rev.1.0_01July2008.pdf

[3] http://www.onsemi.com/pub_link/Collateral/AR0130CS-D.PDF

[4] http://www.raspberrypi.org/

[5] http://www.arduino.cc/es

[6] http://arduino.cc/es/Main/Software

[7] http://botscience.wordpress.com/2012/06/05/historia-de-arduino-y-su-nacimiento

[8] https://xively.com

[9] http://www.cisco.com/web/about/security/intelligence/iot_framework.html

[10] http://es.wikipedia.org/wiki/Sistema_global_para_las_comunicaciones_m%C3%B3viles

[11] http://ocw.upm.es/teoria-de-la-senal-y-comunicaciones-1/comunicaciones movilesdigitales/contenidos/Presentaciones/GSM-07.pdf

[12] http://www.movistar.es/particulares/coberturas/movil/4G

[13] http://es.wikipedia.org/wiki/Servicio_general_de_paquetes_v%C3%ADa_radio

[14] http://www.cooking-hacks.com/index.php/catalogsearch/result/?q=gprs

[15] http://arduino.cc/es/Guide/Environment

[16] http://arduino.cc/es/Main/Software

[17] http://www.elecrow.com/download/SIM900_AT_Command_Manual_V1.03.pdf

[18] http://water.usgs.gov/edu/watercyclecatalanhi.html

Llibres:

ÓSCAR TORRENTE ARTENGO. *Arduino: Curso práctico de formación*
España: Rc Libros, 2013. 588 p. ISBN 978-84-940725-0-5

MASSIMO BANZI. *Introducción a Arduino*
España: Anaya Multimedia, 2012. 128 p. ISBN 978-84-415-3177-2