Universitat Autònoma de Barcelona

ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA

Estudi per a un sistema de mesura de pes per a marmota alpina

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per Josep Mir Izard
i dirigit per
Joan Oliver
Bellaterra,1 de juny de 2009

Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria

El sotasignat, Joan Oliver Malagelada

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en Josep Mir Izard.

I per tal que consti firma la present.

Signat: .....

Bellaterra, 15 de Setembre de 2009

Índex

1. INTRODUCCIÓ

Aquest projecte de final de carrera tracta sobre viabilitat de la construcció d'un sistema per al seguiment del pes d'una població de marmotes en alta muntanya. Es construirà una bàscula, de manera que en ser trepitjada per una marmota, automàticament s'activarà i n'enregistrarà el pes.

Els objectius del projecte són poder enregistrar durant un període que va d'uns dies fins cap a un mes una sèrie de mesures que aportaran sengles sensors. Així doncs, com que el que es vol és controlar l'evolució d'una població de marmotes la mesura principal és el pes. La segona mesura a registrar és la temperatura. Això permetrà fer correlacions entre el comportament de les marmotes i aquesta variable. També es guardarà el temps en què s'han enregistrat les diferents mesures.

La bàscula està basada en un sensor de força del tipus Strain Gauge de la marca Futek, que pot mesurar un pes de fins a 9 kgs, amb sortida de voltatge diferencial.

Aquesta sortida analògica es connecta a un microcontrolador ATmega8 que, mitjançant un algorisme desenvolupat en aquest projecte, està contínuament en escolta fins a detectar un canvi sobtat en el pes.

La modularitat del sistema permet que a més de la temperatura puguin enregistrar-se altres dades.

El microcontrolador és el component principal, el que fa la bàscula *intel·ligent*. Integra tots els diferents dispositius i n'enregistra els resultats ambientals com la pressió, la lluminositat, la temperatura... i la mesura principal: el pes.

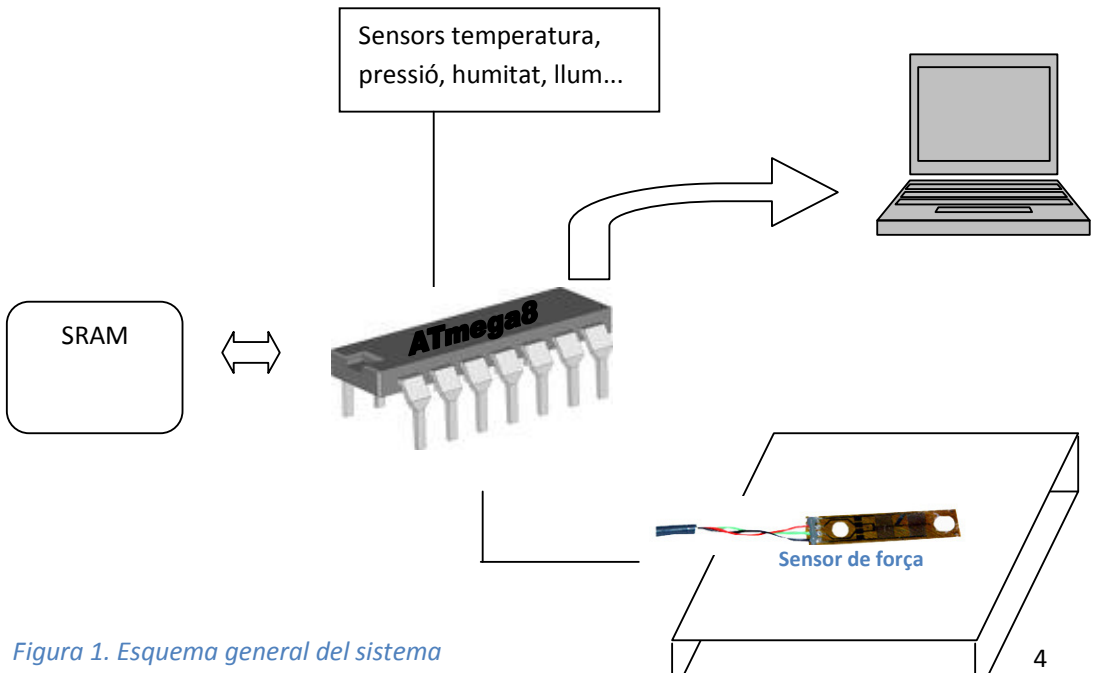


Figura 1. Esquema general del sistema

El microcontrolador ATmega8 de la marca Atmel ha estat escollit per aquest projecte per tractar-se d'un dispositiu que es creia que tenia suficient potencia computacional. Disposa de 8Kb de memòria Flash per a codi intern, d'1Kb de memòria SRAM interna, conversors d'analògic-digital, comunicacions RS-232, i 3 ports de comunicació. Tot i així, més tard s'ha vist que aquest microcontrolador es queda curt a l'hora de gestionar la memòria.

La planificació, la qual s'explica en el següent apartat, ha estat dividida en 4 fases (anàlisi de requeriments, disseny i construcció, proves, i documentació). Tot i així, és una planificació orientativa realitzada en iniciar-se el projecte, de manera que els terminis han estat una mica flexibles.

2. PLANIFICACIÓ

Podem dividir tota la planificació en **4 fases**. Aquests passos s'han anat fent segons la planificació temporal del projecte mostrada a la Figura 2. A continuació s'explica en què consisteixen cada una d'elles.

- **2.1 Anàlisi de requeriments**

En aquesta primera fase es recullen tots els requisits que ha de complir el sistema. Podem dividir aquests requisits en requisits hardware i software. Els hardware inclou: les condicions de treball del sistema, ja que funcionarà a la intempèrie. Una comparativa de les possibles arquitectures del microcontrolador i la tria de la més adequada. També un anàlisi dels possibles sistemes de pes i la tria del sistema de pes més adequat. Finalment, pel que fa al hardware, es proporciona l'esquema elèctric que fa possible el sistema.

Pel que fa al software, s'argumenta la tria del llenguatge de programació, i el disseny dels dos programes de la part de l'ordinador, així com perquè s'ha decidit separar el programa en dos.

La fita a aconseguir és disposar de tot el material necessari per a la construcció del sistema.

- **2.2 Disseny i construcció**

Com el nom indica, és la construcció del sistema prèviament estudiat. Una vegada més, separem aquesta fase en dos parts, la hardware i la software.

En primer lloc estudiem els diferents modes d'energia del hardware, i n'optimitzem el funcionament per a un estalvi d'aquesta. A continuació, la part més extensa és la del disseny d'un protocol per guardar les dades a la RAM de la manera més eficient possible (optimitzar l'escassa memòria de què disposem), i la forma com transmetem aquestes dades per la interfície RS-232. La fita a aconseguir és tenir el sistema implementat i en funcionament. Poden haver-hi petits errors que es poliran a la fase de proves.

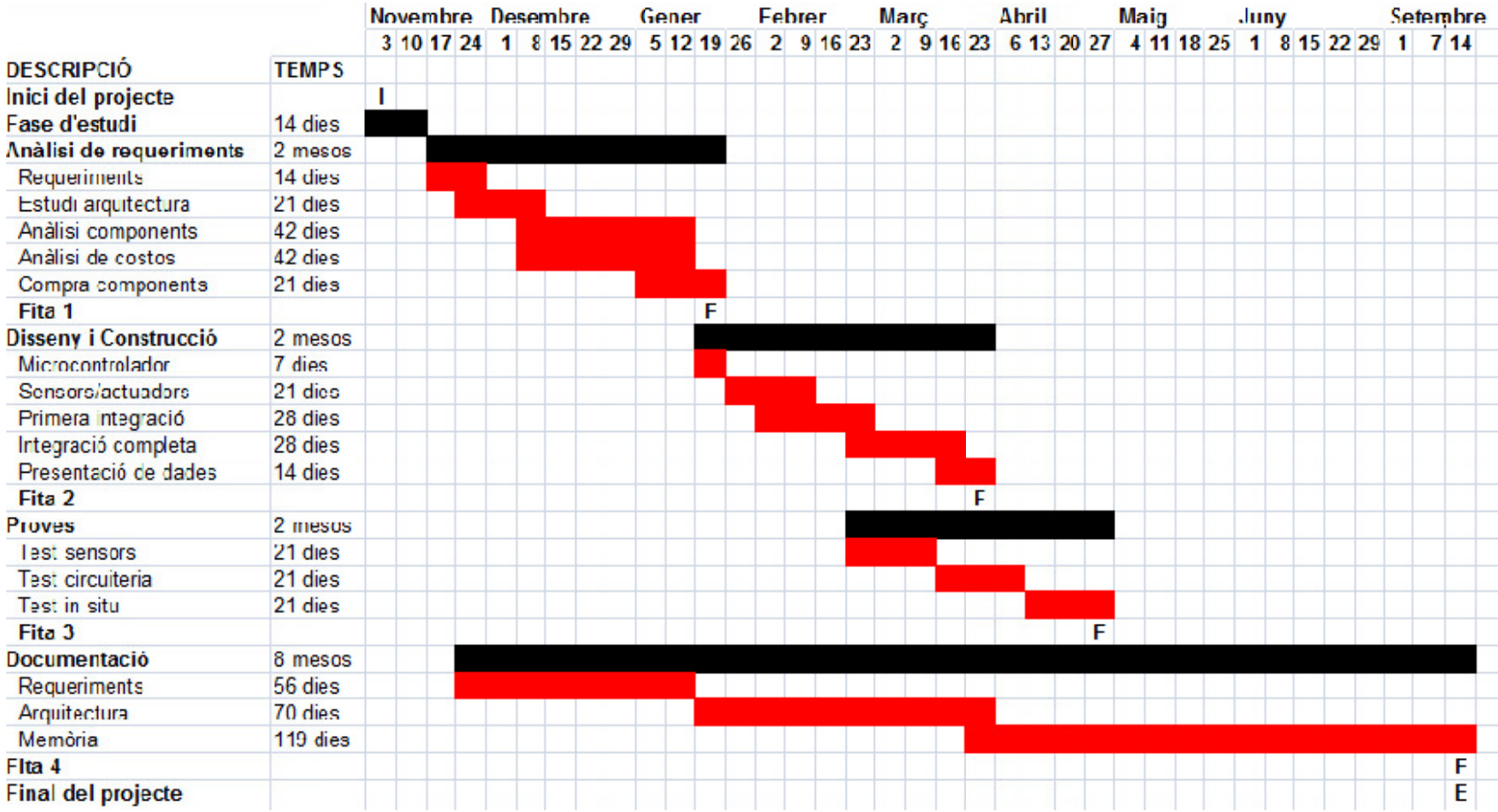
- **2.3 Proves**

Durant la construcció del sistema es van fent vàries proves. La principal és la que es fa al final i es la comunió de totes: connectar la bàscula, provocar canvis de pes i temperatura, baixar-ho a l'ordinador i analitzar-ho amb el programa. Les proves han estat incrementals. Es va començar amb un programa que encenia i apagava un LED. Després un programa que enviava caràcters pel port sèrie. Després que ho fés cada segon, per comprovar els timers. Etcètera. La fita a aconseguir és que el sistema funcioni sota els paràmetres establerts.

- **2.4 Documentació**

La documentació consta de vàries parts. La que es fa prèvia al projecte (memòria prèvia) i inclou, a més d'una introducció, un esquema amb els diferents blocs que inclou el sistema, un anàlisi de l'estat de l'art, un estudi de viabilitat, i conclusions.

Després hi ha la que es fa posterior al projecte (tot i que és bo anar recollint impressions i esborranys durant les fases anteriors) que és la memòria. Aquesta ha de ser la documentació principal. Els seus apartats principals són l'anàlisi de requeriments i de costos, la planificació, el disseny i la construcció del sistema en sí, i per últim els resultats aconseguits i les possibles millores i/o ampliacions del sistema en estudi. La fita a aconseguir és tenir tot el sistema documentat a la memòria.


Novembre Desembre Gener Febrer Març Abril Maig Juny Setembre
3 10 17 24 1 8 15 22 29 5 12 19 26 2 9 16 23 2 9 16 23 6 13 20 27 4 11 18 25 1 8 15 22 29 1 7 14
DESCRIPCIÓ
TEMPS
Inici del projecte
I
Fase d'estudi
14 dies
Anàlisi de requeriments
2 mesos
Requeriments
14 dies
Estudi arquitectura
21 dies
Anàlisi components
42 dies
Anàlisi de costos
42 dies
Compra components
21 dies
Fita 1
F
Disseny i Construcció
2 mesos
Microcontrolador
7 dies
Sensors/actuadors
21 dies
Primera integració
28 dies
Integració completa
28 dies
Presentació de dades
14 dies
Fita 2
F
Proves
2 mesos
Test sensors
21 dies
Test circuiteria
21 dies
Test in situ
21 dies
Fita 3
F
Documentació
8 mesos
Requeriments
56 dies
Arquitectura
70 dies
Memòria
119 dies
Fita 4
F
Final del projecte
E

3. ANÀLISI DE REQUERIMENTS

Es vol una bàscula que enregistri el pes de marmotes quan aquestes hi passin per sobre. A més, periòdicament es prendran mesures d'altres variables com la temperatura.

Com que les marmotes habiten a l'alta muntanya, la bàscula haurà de treballar en **condicions climatològiques** més extremes que les normals, i també en les pròpies de la intempèrie: pluges, vent... El seu funcionament però, no és necessari en temperatures properes als 0 graus i sota zeros, ja que les marmotes tenen un període d'hibernació d'uns sis o set mesos a l'any, des de mitjan octubre fins al maig, amb el qual la bàscula s'estalviarà el funcionament en els mesos més freds de l'any.

Pel que fa a la naturalesa dels **elements a pesar**, cal dir que les marmotes rarament superen els 8 kg de pes (les femelles fan entre 2,8 i 4,5 kg, mentre que els mascles acostumen a fer entre 3 i 8 kg). Així doncs, el sensor de força que incorporem a la bàscula no cal que mesuri més enllà d'aquest pes.

Cal tenir en compte que es desitja un sistema d'**autonomia amplia** (un mes mínim), ja que es pensa en un sistema autònom on la recollida de dades i possibles posades a punt es faran només de forma espaiada en el temps. Sobretot degut a les condicions d'accés de la muntanya. Així doncs, és necessari optimitzar tant els requisits d'energia del sistema com les necessitats de memòria d'aquest.

Per últim, la **descàrrega de dades** es realitzarà amb un dispositiu mòbil (des d'un ordinador portàtil a una PDA), pel que el programa de descàrrega de dades cal que sigui amb una interfície simple, de poc processament, i de mida petita. A més, si aquesta descarrega es realitza de manera inalàmbrica, això facilitarà certes tasques, com ara el no haver de desenterrar la bàscula del terra.

3.1 Requeriments Hardware

- **3.1.1 Arquitectura microcontrolador**

  Hi ha vàries raons per l'ús d'un microcontrolador: l'orientació cap a entrades/sortides enfront el processament intensiu (enfront DSP), la facilitat de modificació per software del programari (vers programació hardware com FPGA), i el seu tamany reduït, baix cost i poca alimentació energètica (front altres plataformes hardware molt més potents com ara una PDA, un telèfon mòbil, un PC..). Usar una FPGA comporta uns dispositius més cars, i una programació no estàndard. Com que estem fent un prototipus, ens interessa un dispositiu senzill i fàcil de programar per fer proves.

El processador que necessitem ha de tenir entrades de senyal analògiques i digitals, i ha de tenir varis pins de sortida digital per poder controlar la memòria SRAM. Aniria bé que tingués suport per a busos de comunicacions com el RS-232 o l'USB. També és important que tingui un mode sleep per a l'estalvi d'energia, així com timers interns que despertin el microcontrolador al cap d'un cert temps. Ha de ser fàcilment prototipable i tenir un baix cost. Per últim, ens interessa que tingui una freqüència de rellotge suficient per a l'enregistrament d'uns esdeveniments que es poden donar molt ràpid en el temps.

A continuació fem una petita comparativa dels diferents microcontroladors que s'han tingut en consideració per implementar el nostre sistema:

1.**Texas Instruments** MSP430F1122 8MHz Micro. 16-bit 4k bytes Flash Memory
-3 ports de 8 bits
-ADC de 10 bits
-comunicació USART
$4.90 – Referència MSP430F1122IDW a futurlec.com

2.**Atmel** ATMega8L 28-Pin 8MHz 8kb 8-bit Microcontroller (RoHS Compliant)
-2 ports de 8 bits i un de 7 bits (total 23 bits)
-6 canals ADC amb resolució 10 bit
- possibilitat de programació en C
-comunicació USART
$1.50 –a farnell.com

3.**Motorola** 68HC11 4MHz 1kb 8-bit Micro. with A/D (20 pin DIP)
-4 ports de 8 bits i un de 6 bits (total 38 bits)
-Fins a 8 canals ADC amb resolució 8 bit, que esta al Port E (es resten)
-Stop Mode (sleep) amb Auto Wakeup cada cert temps establert
10.19€ –a farnell.com

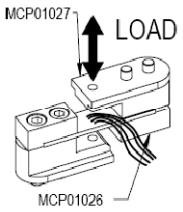
Les característiques de treball del microcontrolador són prou genèriques com per a què qualsevol d'aquests micros vagi bé. Per facilitat de treball i preu s'ha decidit emprar el ATMega8.

Com que només s'està mirant la viabilitat del sistema, l'escassetat de pins d'entrada/sortida de l'ATmega8 no és un problema, ja que les games superiors (com l'ATmega128), q tenen la mateixa arquitectura, compleixen de sobres els requeriments de capacitat que necessito (fins a 6 Ports amb aprox. 8 sortides cada un).

- **3.1.2 Sistema de pes**

El pesatge de la marmota es realitzarà emprant un sensor de força. En el mercat es disposa de varis sensors de força, de diferents tecnologies, característiques, i preus diferents. Entre les diferents possibilitats analitzades hi trobem els següents sensors:

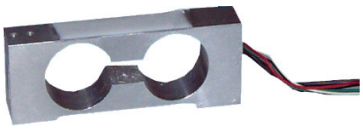
Strain gauge de Futek



Un strain gauge és un element que porta una resistència implantada a la superfície. Quan la superfície es corba per l'acció d'una força externa, l'àrea s'eixampla creant més resistència. El sensor en qüestió de la marca Futek és de molta precisió. Pot pesar fins a 20 lliures (9 kilos). S'alimenta i treu una sortida en forma de voltatge diferencial. (més informació a [1])

Item FSH01455 de Futek [2] - $90

Load Cell de Elane



Load cell de fins a 10kg amb alimentació. Una load cell és un element més precís que un strain gauge, i en aplicacions industrials, aquests acostumen a ser molt més cars. Una load cell està formada per 4 strain gauges. (més informació a [3])

Load cell 10 kg de Scales-r.us [4] - $9

Sensor de pes de Robotshop - Interlink Electronics 0.5" Circular FSR

Tot i que com indiquen es tracta més d'un element per a fer mesures qualitatives més que no pas de precisió, és un producte indicat pel seu baix preu ($6.60). No es tracta ni d'un load cell ni d'un strain gauge, sinó d'un aparell que porta un "gruixut film de polimer" que decrementa la resistència a més pressió.

Producte RB-Int-02 de robotshop.ca [5] - $6.60

Finalment s'ha escollit el strain gauge de la marca Futek, ja que era el que estava disponible inicialment. Per a un desenvolupament futur s'hauria de comparar el seu funcionament amb la load cell d'Elane, més econòmica i que sembla també satisfà els requeriments.

- **3.1.3 Electrònica del sistema**

El strain gauge fb3300 de Futek es pot alimentar amb un voltatge de fins a 18 Volts. El sensor retornarà 2mV per cada Volt nominal. Així, com que nosaltres l'alimentem amb 5V, el sensor pot retornar fins a 10mV. Aquest retorn és en forma de voltatge diferencial. És a dir, per una banda tenim dos cables per l'alimentació del sensor, i per l'altre els dos cables per on es dona una diferència de tensió.

Les especificacions tècniques del strain gauge són aquestes:

RATED OUTPUT	2 mV/V nom.
SAFE OVERLOAD	150% of R.O.
ZERO BALANCE	±5% of R.O.
EXCITATION (VDC OR VAC)	18 MAX
BRIDGE RESISTANCE	1200±300 Ω
INSULATION RESISTANCE	1000 M ohms @ 50 VDC
COMBINED ERROR	±0.25% of R.O. \ ACTUAL LEVEL OF ACCURACY
NONREPEATABILITY	±0.05% of R.O. } DEPENDS ON CLAMPING METHOD.
TEMP. SHIFT ZERO	±0.02% of R.O./°F [0.036% of R.O./°C]
TEMP. SHIFT SPAN	±0.02% of LOAD/°F [0.036% of LOAD/°C]
COMPENSATED TEMP.	60 to 160°F [15 to 72°C]
OPERATING TEMP.	-60 to 200°F [-50 to 93°C]
MATERIAL	300 SERIES S.S.
WEIGHT	0.35 oz [10 g]
DEFLECTION	0.004 to 0.010 [0.10 to 0.25] nom.

CABLE: #28 AWG, 4 Conductor, Spiral Shielded Silicone Cable 1 ft [0.3 m] Long

A continuació hem d'amplificar el senyal, ja que en ser de miliVolts no pot ser llegit directament pel microcontrolador. La senyal final volem que estigui en un rang aproximat de 0-5V, el rang que tenen els conversors ADC del microcontrolador.

Per amplificar el senyal fem servir el circuit integrat INA2126, un amplificador d'instrumentació de precisió, que internament estan basats en dos amplificadors operacionals i que se serveixen d'una resistència externa per graduar el coeficient d'amplificació.

- **3.1.4 Comunicació micro-PC**

La comunicació amb l'ordinador per a la descàrrega i anàlisi de dades es fa amb el protocol RS-232. Aquest protocol ja es troba implementat dins del microcontrolador ATmega8, facilitant-ne molt la seva implantació. Un inconvenient d'aquest sistema és que avui en dia els ports sèrie i en paral·lel són cada vegada més obsolets, amb el qual pot ser necessari la instal·lació d'algun adaptador tipus USB-port sèrie.

Tot i que per aquest projecte s'ha descartat, podria ser molt interessant l'ús d'un adaptador Bluetooth-Port sèrie. Aquest adaptador, que estaria inserit a la bàscula, faria

que no l'haguéssim de "desenterrar" del lloc on es troba i poder descarregar les dades disposant només d'un aparell amb bluetooth, amb el que ens estalviem el anteriorment citat problema de trobar aparells que disposin dels cada vegada més escassos ports sèrie. Un exemple seria el Bluemore 200 [6], de 45 €.

La programació del microcontrolador, que només cal fer en el procés de desenvolupament, es fa a través del port paral·lel mitjançant la utilitat PonyProg [7].

3.2 Requeriments Software

- **3.2.1 Llenguatge de programació**

A la part del **microcontrolador** teníem vàries opcions pel que fa al llenguatge de programació. Una és en el llenguatge nadiu del microcontrolador: l'ensamblador. Amb força documentació però de més difícil comprensió, més lent de programar i més difícil de mantenir. Tot això juntament amb la falta de destresa en aquest llenguatge per part de l'autor del projecte va fer descartar aquest llenguatge.

L'altra opció és en el llenguatge C, tot i que s'ha de dir que més limitat de l'habitual com és obvi a l'estar treballant en un entorn encastat. L'entorn AVR disposa d'un *port* del compilador de codi obert GCC que genera un codi ensamblador, diuen, de gran qualitat. Les qualitats intrínseques de tot llenguatge de programació d'alt nivell feren que ens decantéssim per aquest llenguatge.

Així doncs, la compilació del codi del microcontrolador la fem amb l'entorn AVR Studio 4, de la pròpia casa Atmel. Una vegada tenim el codi compilat, aquest es passa al microcontrolador amb el programa PonyProg.

3.3 Anàlisi de Costos

El cost del projecte ve donat en la seva major part pels dispositius electrònics i sensors que fem servir. No es contemplarà en aquest cas el cost de la mà d'obra, el de l'equipament electrònic (multímetre...) ni els de l'adquisició del software de desenvolupament (Microsoft Visual Studio...).

Hardware	
ATmega8	1€
Memòria SRAM	5€
Sensor de força Futek	63€
Suport sensor de força Futek	28€
3 protoboards	24€
Resta components electrònics	20€
TOTALS	141€

4. DISSENY I CONSTRUCCIÓ

4.1 Hardware

4.1.1 Sistema de mesura

Per la part hardware del sistema, s'han emprat diferents component electrònics i circuits integrats, explicats breument més avall. Per muntar el sistema s'han emprat un parell de plaques Protoboard per fer un prototipatge ràpid (veure figura 3), ja que és un sistema en que no cal soldar i on es poden canviar les connexions ràpidament. Per a un sistema definitiu, s'hauria d'emprar una configuració més robusta com ara soldar-ho, o fins i tot usar una placa de circuit imprès (Printed Circuit Board).

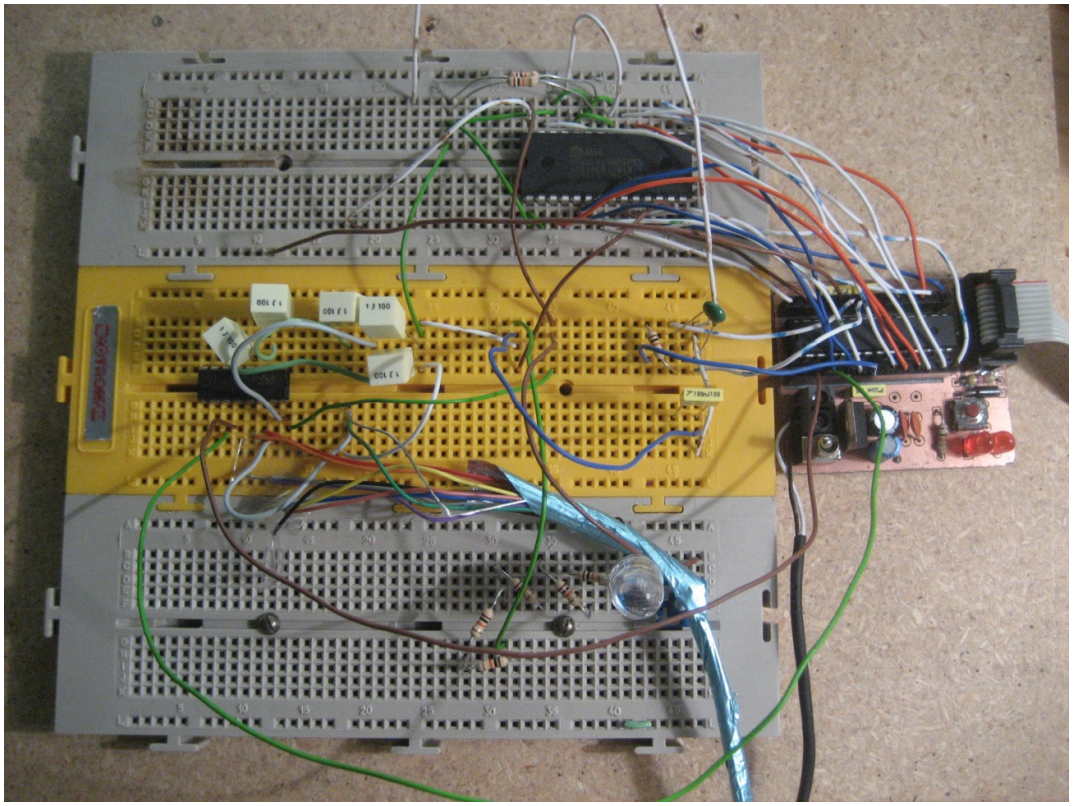


Figura 3. Electrònica del sistema

En la figura 4 es mostra l'esquema de l'electrònica del sistema:

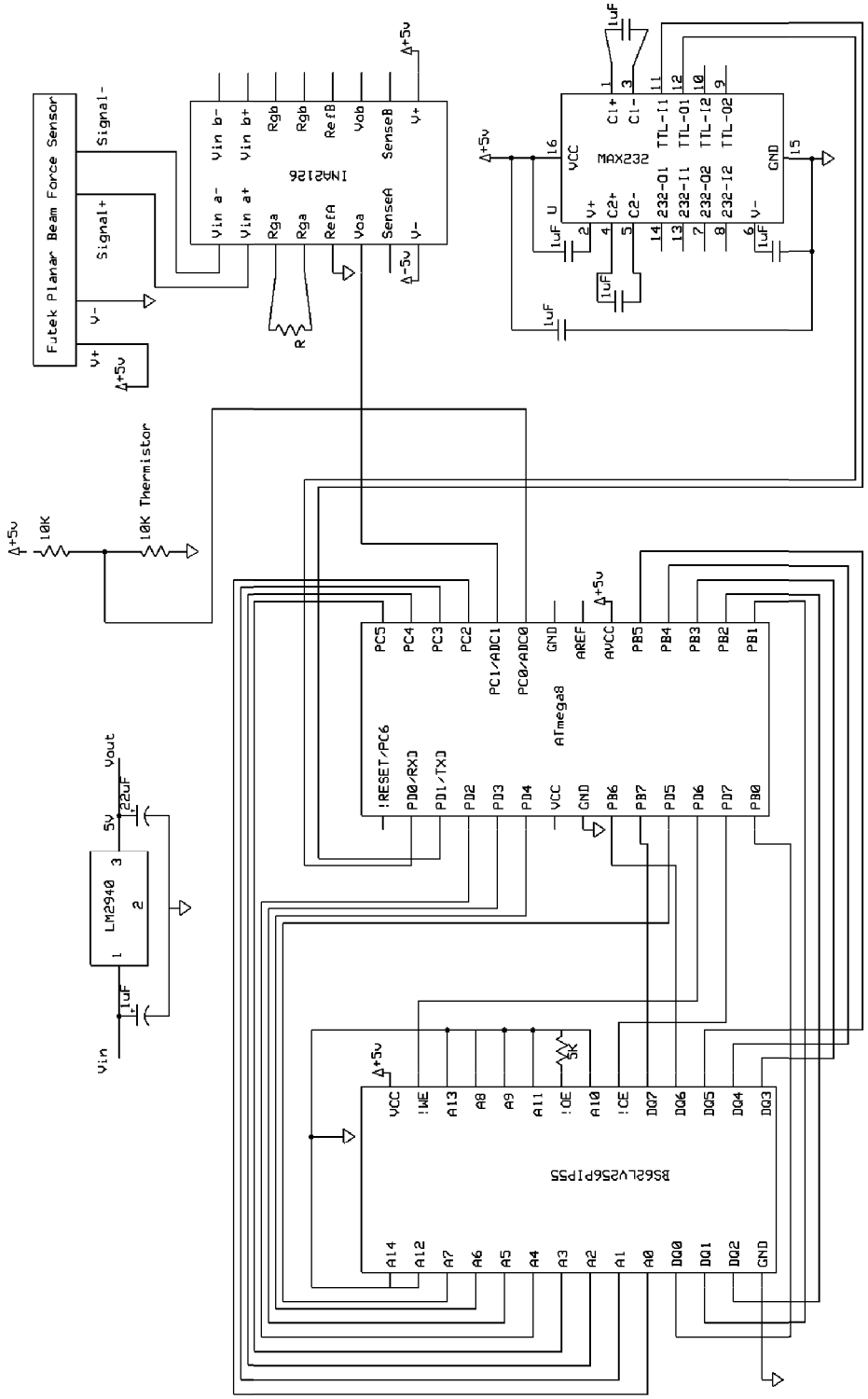


Figura 4. Esquemàtic de l'electrònica del sistema

Els components principals del prototipus són:

ATMEGA8

És la part central del sistema. Dels seus diferents mòduls, els que aprofitem són dos timers, dos conversors A/D, el mòdul de comunicació USART, i els seus ports com a entrades/sortides digitals.

El pin 6 del port C no es pot fer servir amb el nostre prototipus, ja que hi ha un fusible, que no es pot canviar una vegada programat, que estableix si el pin es comporta amb una funció (RESET) o una altra (Entrada/Sortida). En el nostre cas aquest fusible ha estat programat com a RESET, de manera que ara no el podem usar com a E/S. Tot i així, si es canviés el xip per un de nou, i canviant una mica el codi, sí que podríem usar aquest port com a E/S, ja que el RESET no el fem servir.

MAX232

Aquest circuit integrat fa de pont entre dos senyalitzacions diferents. Del sistema TTL/CMOS (implementat als ports sèrie i paral·lel de l'ordinador) que usa voltatges de 12V, al sistema del RS232 que usa el microcontrolador, amb voltatges de 5V. En cas d'emprar un adaptador bluetooth com el descrit més a munt als requeriments, aquest component seria irrellevant, ja que l'adaptador bluetooth pot funcionar directament amb el protocol UART sense haver-lo de passar a RS232.

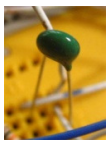
Amplificador d'instrumentació - INA2126

Aquest circuit per a instrumentació amplifica la senyal que surt del strain gauge, ja que aquest és de l'ordre del miliVolt. El guany s'estableix amb el valor d'una resistència que es connecta als seus pins.

Regulador de tensió - LM2940

Regulador de tensió. S'assegura que posem el voltatge que posem d'entrada (fins a uns 26 V), el voltatge de sortida sigui 5V. És a dir, a Vin connectarem l'alimentació del sistema, i a Vout tots els pins on hi ha marcat que han d'anar a 5V.

Termistor de 10K



Resistència variable en funció de la temperatura (termistor). Partint d'una resistència de 10K, a més temperatura, menys resistència. Juntament amb una altra resistència de 10K, es forma un divisor de tensió que a més temperatura

donarà més voltatge.

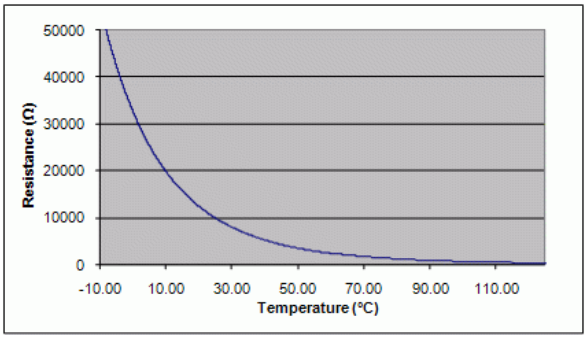
Les característiques d'un termistor es poden determinar amb l'equació de **Steinhart-Hart**.

$$\frac{1}{T} = A + B*\ln(R) + C*\ln(R)^3$$

on

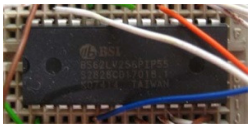
A = 3.354016 E-03
B = 2.56985 E-04
C = 6.383091 E-08
R = resistència del termistor, en ohms

Aquesta equació dóna la següent gràfica, que mostra el canvi de resistència en funció de la temperatura:



Per últim, dir que el nostre termistor opera dels -30º als 125º.

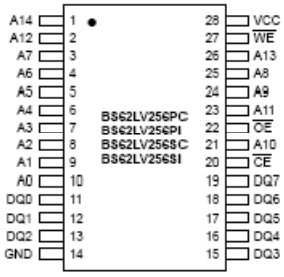
Memòria SRAM - BS62LV256



Aquesta memòria disposa de Chip Enable per estalviar energia quan no s'utilitza, a més dels Write Enable (per indicar lectura o escriptura) i el Output Enable, que sempre estarà actiu. Al ser Static RAM implica que sempre hi ha d'haver un cert voltatge per tal que les dades es mantinguin.

Els pins de la memòria que fem servir són els 8 del byte de dades, 2 dels 3 bits de control (el pin Output Enable no el necessitem), i 8 pins per l'adreçament de la memòria, suficients per les simulacions del nostre estudi.

Així doncs, amb 8 bits de per direccionar la memòria, podem direccionar 2^8 posicions. Tenint en compte que a cada posició hi ha 1 bytes, ens resulten 256 bytes. És una quantitat totalment simbòlica, ja que el micro mateix ja ofereix 1 kb de SRAM interna. Però el propòsit de muntar



aquesta configuració és deixar tots els mecanismes enllestits de manera que el sistema sigui fàcilment escalable, i només actualitzant a un microcontrolador amb més ports i amb algunes petites modificacions, fàcilment puguem arribar a direccionar els 32kb de la nostra memòria i fins i tot memòries més grans.

Esquemàticament doncs:

-8 pins pels 8 bits del port de dades

-2 pins de control, d'un total de 3. El Output Enable sempre estarà actiu.

Write Enable – per seleccionar si és una operació d'escriptura o de lectura

Chip Enable – quan no estigui seleccionat la memòria estarà en standby (estalvi d'energia)

Output Enable – serveix per tallar o no la sortida de dades. Sempre podrà estar actiu (LOW)

-8 pins d'adreça, que permeten direccionar 512 bytes

4.1.2 Alimentació

Els elements principals que consumeixen son el microcontrolador i la memòria. Com que disposen de modes d'estalvi d'energia, a continuació s'explica les mesures que s'han pres per tal d'estalviar-ne, així com els diferents gastos energètics de cada estat.

El **microcontrolador**, a una freqüència de 4 Mhz, 3V i 25ºC consumeix, segons el mode:

– Active: 3.6 mA
– Idle Mode: 1.0 mA
– Power-down Mode: 0.5 µA

	Active Clock Domains					Oscillators		Wake-up Sources					
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Osc. Enabled	INT1 INT0	TWI Address Match	Timer 2	SPM/ EEPROM Ready	ADC	Other I/O
Idle			X	X	X	X	X(2)	X	X	X	X	X	X
ADC Noise Reduction				X	X	X	X(2)	X(3)	X	X	X	X	
Power Down								X(3)	X				
Power Save					X(2)		X(2)	X(3)	X	X(2)			
Standby(1)						X		X(3)	X				

Figura 5. Funcions disponibles segons el mode de treball del microcontrolador

Es descarta el mode **power down** ja que no podríem tenir els timers que serveixen per controlar l'hora i el termòmetre. Sí que es mantenen les interrupcions externes, per tant hauríem de despertar el micro amb una interrupció externa per exemple amb un circuit 555 (un timer extern). Però per aquest projecte s'ha descartat per ser massa complex. A més també s'ha de tenir en compte que tot i que el microcontrolador estalviaria energia, tindríem un nou component a alimentar.

El mode **standby** és idèntic al power-down excepte que l'oscil·lador continua funcionant.

En mode **power-save** es pot usar el timer2 (el 3r dels 3 que hi ha). El problema és que per fer-ho s'ha de fer servir un clock extern en els ports TOSC1 i TOSC2. I hem de tenir en compte l'escassetat de ports, sobretot per la gran quantitat de ports que la memòria fa servir.

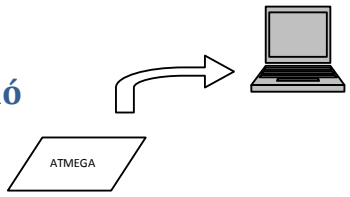
Així doncs, la millor alternativa que tenim és la d'usar el mode **idle**, que com a mínim fa servir quatre vegades menys corrent que en mode normal (de 3,6 mA a 1,0 mA).

Per optimitzar el consum també hem d'apagar els mòduls que no utilitzarem. En el nostre cas, per software deshabilitem **l'Analog Comparator** i el **Watchdog timer** (sistema per controlar que el microcontrolador no està penjat). El **Brown-out Detection** també es podria desactivar si volguéssim estalviar energia, però en el nostre cas no podem ja que no es programa per software sinó per fusible. Apart de que és una característica que s'ha mostrat útil alguna vegada, com quan la RAM necessitava massa corrent, fent baixar el Voltatge del micro, amb el qual observava un Reset provocat per aquest sistema de seguretat.

La **memòria** és de tipus *very low power consumption*. En mode normal consumeix 1.5mA a 1Mhz (més o menys la velocitat a que funciona el micro), mentre que en mode stand-by només consumeix 0.4uA.

Hi ha la possibilitat de fer funcionar aquesta memòria a una tensió inferior de 3V cas que es disposi del regulador adequat.

4.2 Comunicació



El principal paràmetre que s'ha tingut en compte a l'hora de triar el format de les dades ha estat el d'optimitzar l'espai que ocupaven, per tal d'augmentar els dies que la bàscula pot mantenir-se sense intervenció, i també per agilitzar el procés de descàrrega de dades per port sèrie. Tot i així, tampoc s'han sobreoptimitzat, comprimint-les i demés, cosa que haguera augmentat la complexitat del programari.

Hi ha dos tipus de mesures a enregistrar, la temperatura i el pes. Aquestes mesures sempre aniran acompanyades per un altre element, l'horari. Així doncs, sempre que fem una mesura sabrem alhora a quin temps s'ha pres.

Les dos mesures tenen criteris diferents per a ser enregistrades. La temperatura es pren de forma periòdica cada 30 minuts (en el nostre prototip s'ha fet que sigui de 30 per poder mostrar el funcionament sense haver d'esperar grans estones de temps).

La mesura de pes no és periòdica, sinó que només s'enregistrarà quan es passi d'un cert llindar. En el nostre cas hem triat un llindar de 1,5 kgs, per tal de descartar la terra amb què es pot cobrir la bàscula, possibles pedres, o en definitiva qualsevol element que no sigui la marmota. Així ens estalviarem d'enregistrar pesos que no siguin una marmota. (No està de més recordar que aquestes acostumen a pesar un mínim de 2.8 quilos, pel que el llindar de 1,5kg no té perquè afectar el bon funcionament).

Durada

Com que la memòria és finita, i el sistema, encara que no enregistri pes, enregistra temperatura periòdicament, podem deduir que el nostre sistema tindrà una durada. Si la durada és de, posem, 6 mesos, una vegada estiguem al 7è mes, les dades del 1r mes es començaran a sobreescriure (la memòria funciona de manera circular).

Com que sabem la periodicitat dels enregistraments de temperatura, podem fer una estimació de la durada màxima, això és, suposant que no es dóna cap mesura de pes. Com que les mesures de pes que hi haurà són imprevisibles (depèn de les vegades que l'animal surti del cau), no podem fer més que una estimació suposant unes quantes mesures de pes per dia.

Quan es tingui el sistema completament implementat amb el xip ATmega128 i per tant amb possibilitat d'adreçar tot l'espai de la memòria (32kb), tindrem una durada de més de 3 mesos.

32768 bytes * 30 minuts/7 bytes= 140434 minuts = 3 mesos i 7 dies

Suposant un cas més realista, on posem que hi haurà 12 enregistraments de pes cada dia, tenim una durada de:

(12 enregistraments al dia són una mesura de pes cada 2 hores. En dos hores per tant hi haurà 7bytes del pes + 7bytes*4 de la temperatura = 35 bytes)

32768 bytes * 2 hores /35 bytes = 78 dies = 2 mesos i 7 dies

De moment, amb el nostre prototip de 256 bytes, la durada que tenim, tenint en compte una periodicitat de temperatura de 30 segons, i sense que hi hagi mesures de pes, és de 18 minuts.

Un temps adequat ja que és aproximadament el que pot ser útil per fer proves. Amb això puc tenir una primera visió de com pot funcionar el sistema.

256 bytes * 30 segons/7 bytes = 18 minuts

Memòria per l'horari

H			
255	hora	minut	segon

Cada posició representa un byte. El primer byte informa del contingut del "paquet de dades". S'ha triat un nombre alt perquè no es pugui confondre amb una dada del contingut (un segon mai arribarà a 255, sinó només a 60). Així, la memòria necessària per emmagatzemar el temps és de 4 bytes.

Memòria necessària per termòmetre:

T		
254	part1	part2

Suposant que desarem un decimal, i que el rang de temperatura que volem abarcar és de 0 graus a 40 graus, es tracta de 400 possible números. Donat que un byte només són 256 possibles números, necessitaríem 2 bytes per cada mesura de temperatura (65535 possibles valors).

Aleshores, hem de tenir en compte que aquestes mesures es faran cada 15 minuts. Així tenim que en un dia necessitem 192 bytes, i en un mes uns 6 kbytes.

Memòria necessària per sensor de força:

P		
253	pes1	pes2

El sensor de força té un rang de 0 a 9 kg. Però si tenim en compte que la bàscula ja aplica un pes constant d'uns 2,5 kgs, tenim que hem d'enregistrar un rang de 6,5 kgs. Per tenir una precisió suficient per copsar els canvis de pes en marmotes que es donin al dia a dia, però tenint alhora en compte que a partir d'un cert nivell la precisió ja no es pot afinar més a causa d'elements externs (pedretes..), decidim que amb dos decimals de precisió n'hi haurà prou.

Per tant, el nostre rang és de 650 possibles valors. Com abans amb la temperatura, no en tenim suficient amb 1 byte (256 posicions), però en tindrem de sobres amb 2 bytes (65536 posicions). A partir d'aquí, la suposició més arriscada de fer és la de quantes vegades la marmota passarà per sobre de la bàscula, amb el conseqüent enregistrament de dades: suposem que unes 3 vegades al dia. Això és una quantitat de dades molt inferior a la requerida per la temperatura: 186 bytes al mes seguint aquest patró. Tenim doncs molt marge d'ampliació per una variable que tanmateix és molt imprevisible.

Amb els 10 bits de la variable pes fem una mateixa distribució que amb la variable temperatura, és a dir, 7 bits a pes 2 i 3 bits a pes1.

Exemple

Així, amb tot el que hem dit abans, posem un exemple del que seria un fitxer de dades descarregat del microcontrolador:

```
251 253 4 52 255 0 0 39 253 7 80 255 0 0 40 254 4 15 255 0 1 0 254 4 10 255 0 2 0 254 4 8 255
0 3 0 254 4 8 255 0 4 0 253 7 127 255 0 4 53 253 7 80 255 0 4 54 253 7 127 255 0 4 55 253 7
80 255 0 4 56 253 7 80 255 0 4 57 253 7 95 255 0 4 58 253 7 80 255 0 4 59 254 4 13 255 0 5 0
253 4 127 255 0 5 0 253 6 48 255 0 5 1 253 4 87 255 0 5 2 253 5 40 255 0 5 3 254 4 16 255 0 6
0 254 4 13 255 0 7 0 254 4 15 255 0 8 0 254 4 15 255 0 9 0 254 4 15 255 0 10 0 254 4 15 255 0
11 0 254 4 12 255 0 12 0 254 4 14 255 0 13 0 254 4 15 255 0 14 0 252 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
250
```

Recordem que del byte 250 en amunt són bytes de control. El 251 marca el inici de la transmissió i el 250 el final. El 252 marca el final de la circularitat, ja que com que la memòria va escrivint-se cíclicament, hem de saber on acaben i tornen a començar les dades. En l'exemple nostre podem veure com encara no s'ha completat cap volta a la memòria, amb el qual els últims bits encara estan a 0 (el valor amb el que s'inicialitzen).

Aleshores, ja entrant en detall, podem veure els diferents paquets de dades. 253 4 52 és una mostra del pes. Tota mostra ve sempre acompanyada d'un paquet de temps, en aquest cas el 255 0 0 39, que significa que ens trobem al segon 39. Aquesta cadena ve seguida d'una altra mostra de pes, amb valor 7 80, que com hem dit, significa 7 * 128 + 80 = 976 i dividit per 10, 9,76 kg.

4.3 Software

4.3.1 Microcontrolador

El programa que porta el microcontrolador segueix una filosofia *event-driven*. És a dir, hi ha una sèrie d'events (SIGNALS) que són els que marquen el ritme del programa. A continuació a la figura 6 es mostra l'estructura del programa.

```
#include ...

SIGNAL (SIG_UART_RECV) {
   accio=1;
}

SIGNAL (TIMER1_OVF_vect) {
   accio=2;
}

SIGNAL (TIMER0_OVF_vect) {
   accio=3;
}

accio=0;

main() {
   → inicialitzem moduls (USART, TIMERS, AD_CONVERTER)
   → desactivem moduls innecessaris per estalvi d'energia (analog comparator, watchdog)

   While(true) {
      Switch(accio) {
         1:
            → envia tota la memoria
         2:
            → augmenta variable temps en un segon
            → si som al segon 30 pren temperatura i enregistra-la
         3:
            → pren mesura de pes
            → si passa d'un cert llindar, enregistra-la
      }

      sleep();
   }
}
```

Figura 6. Pseudocodi del programa del microcontrolador

Les rutines de les interrupcions són tres, i es troben situades a l'inici del programa:

- SIG_UART_RECV : s'activa quan es rep un caràcter via rs-232. Hem programat que si aquest caràcter és l'"s", s'enviï tot el contingut de la memòria.

- TIMER1_OVF_VECT: és una interrupció del timer1 (16 bits), que programem perquè faci un compte enrere amb un número inicial i una freqüència que fa que aquest compte enrere duri un segon. Quan aquest segon ha passat, s'activa la interrupció i nosaltres hem programat perquè s'augmenti la variable segon (i minut i hora si s'escau).

- TIMER0_OVF_VECT: és la interrupció del timer0, de 8 bits i per tant menys precís. Però no és problema, perquè no enregistra temps sinó que serveix per cridar que es faci una mesura de pes. És més freqüent que timer1, per això ho fem en timers separats.

Quan s'executa el programa per primera vegada, no entra dins de cap opció del switch. Per tant passa directament a mode sleep. El microcontrolador sortirà del mode sleep quan es doni una de les 3 interrupcions. Indicarà l'acció que s'ha de fer, i el while la farà. Després tornarà a entrar a mode sleep. Així doncs, el nostre micro està la majoria del temps en mode d'estalvi d'energia, a no ser que un event el reclami i s'activi.

4.3.2 Programes d'usuari

Pels programes que l'usuari final executarà, el de **descàrrega** i d'**anàlisi de dades** , el llenguatge de programació escollit ha estat el C#. Sobretot degut a la familiaritat de l'autor del projecte amb aquest llenguatge i a les eines que la llibreria .NET ja porta implementades. Així doncs, des d'aquest llibreria tant podem accedir fàcilment al port sèrie, com podem representar les dades amb taules i gràfiques. Es tracta d'un llenguatge de compilació intermitja, com el Java, i que com aquest detecta molts errors de programació a mesura que s'escriu el codi, facilitant així la vida del programador. Un inconvenient d'aquest llenguatge és que es necessita un ordinador que usi el sistema operatiu Windows.

Els requeriments del software són dos: descarregar les dades des del micro cap a l'ordinador, i interpretar-les i analitzar-les. Aquestes tasques s'han separat en dos programes diferenciats, per una banda la descàrrega i per l'altra l'anàlisi. Això s'ha decidit així per vàries raons:

Per separar la programació de dues parts clarament diferenciades (divide and conquer), i també l'estalvi de recursos en l'execució, donat que no sempre es necessitaran les dues funcionalitats alhora. Això permet tenir un programa de descàrrega més petit i simple: més fàcilment portable a un dispositiu mòbil com ara una PDA, que es pensa que es farà servir per la descàrrega de dades a l'alta muntanya.

La passarel·la entre els dos programes serà un fitxer de text amb tot el contingut literal de la memòria RAM, és a dir números que van de 0 a 255 i que representen els bytes. L'única operació que s'hi fa és la de reordenar de manera que al principi sempre hi hagi els bytes més antics de la memòria (desfer la circularitat de la memòria), i poc més.

Usar un fitxer intermig permet emmagatzemar les dades de forma senzilla. També editar-les per a propòsits de debugació. I a l'hora de debugar el programa d'anàlisi, podem fer-ho amb un fitxer creat artificialment sense haver d'estar constantment descarregant les dades.

Descàrrega de dades

Com s'ha dit, la descàrrega de dades es fa amb la classe System.IO.Ports.SerialPort ja implementada a la llibreria de .NET. Una vegada tenim el port sèrie de l'ordinador connectat al port sèrie de la bàscula, apretant un botó, s'envia per aquest port el caràcter que fa que el micro iniciï la transmissió. Les dades trameses per la bàscula es visualitzen per pantalla en format byte, és a dir en números que van del 0 a 255. Com es pot veure, aquest programa és molt senzill i compacte, pensat per a ser portat a dispositius mòbils amb pocs recursos. La descàrrega dels 256 bytes dura escassament 10 segons. Després aquestes dades es guarden a un fitxer de text per a que quedin disponibles per al següent programa.

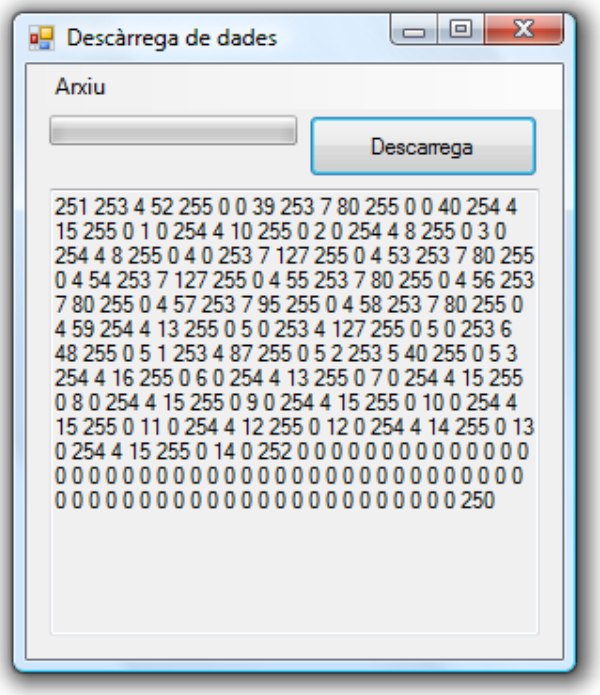


Figura 7. Captura de pantalla del programa de Descàrrega de dades

Anàlisi de dades

Dins el programa s'ha d'obrir un fitxer amb les dades a tractar (que provenen del programa de descàrrega). Aleshores, per cada conjunt de bytes que representa unes dades (hora, temperatura...) es crea un objecte de tipus Dades. Així, l'únic que tindrem serà una llista d'objectes Dades.

Aquestes dades es veuen en un datagrid que diferencia amb el color les mesures de pes i les de temperatura.

També es mostra una representació de les dades en format gràfic, amb dos gràfics superposats, un de barres pel pes i un de línies per les temperatures. Per poder veure aquesta

gràfica és necessari haver instal·lat el Microsoft Chart Controls for Microsoft .NET Framework, ja que és un component que per defecte no ve inclòs amb el framework de .NET .

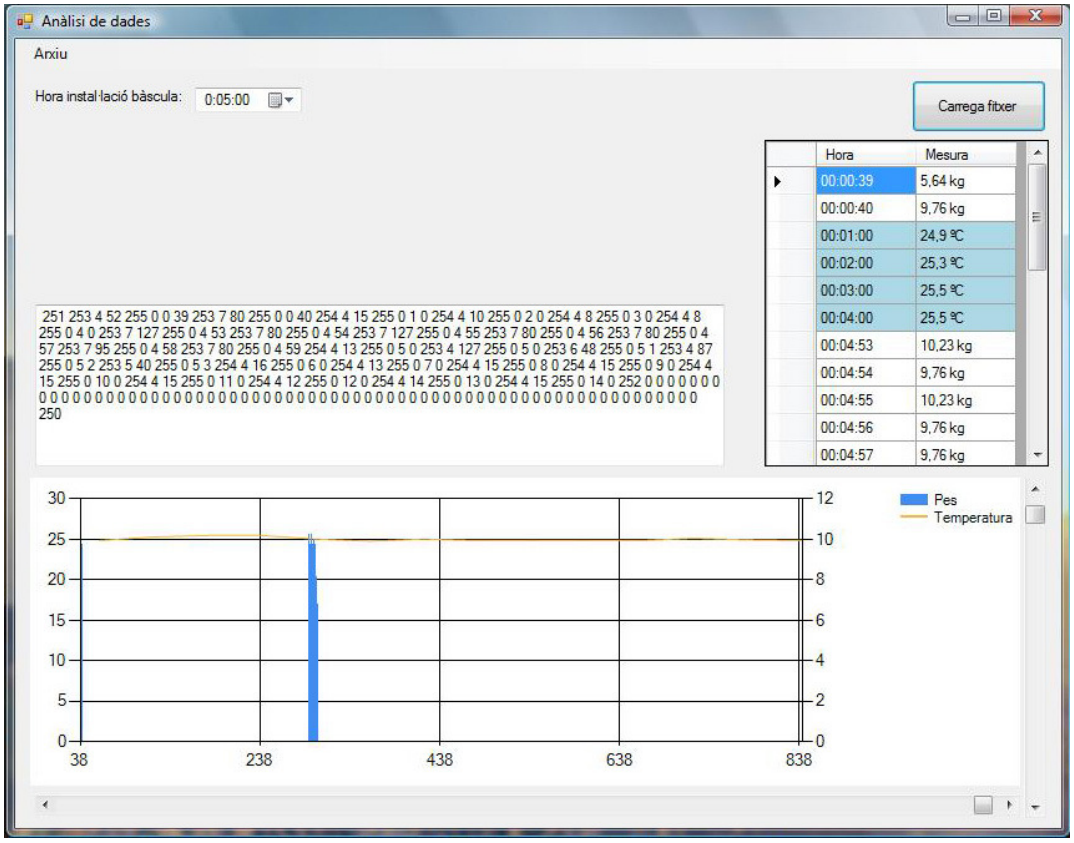


Figura 8. Captura de pantalla del programa d'Anàlisi de dades

4.4 Disseny de la bàscula

L'estructura que ens serveix de bàscula ha estat reaprofitada d'una antiga bàscula de precisió. En un primer moment es va pensar en una construcció pròpia amb fusta. Però en trobar una bàscula antiga que podia ser reaprofitada, es va triar aquesta opció ràpidament, tant per l'estalvi de temps en la construcció, com per que òbviament ja està preparada per pesar coses.

Figura 9. La bàscula. El sensor de força s'instal·la a la barra de fusta

Pel que fa a la **instal·lació en el medi**, la bàscula s'ha pensat per ser posada en un clot de forma quadrada d'uns 50x40cm i d'uns 24 cm de profunditat (les mides de la bàscula) per tal que hi quedi encabida. A nivell de terra passaria desapercebuda (no seria una plataforma on s'hi ha de pujar). A partir d'aquí, seria bona idea cobrir el sot amb alguna làmina de plàstic que tapés les escletxes entre la bàscula i la superfície. Aquesta làmina hauria de ser prou flexible per poder reflectir els canvis de pes que hi haurà a sobre. Aquesta làmina també s'hauria de fixar al terra pels extrems per tal que no es mogués de lloc.

Per últim, seria bona idea cobrir la bàscula amb un plàstic o una bossa el més hermèticament possible per evitar humitat, insectes i altres elements que puguin afectar tant les parts mòbils de la bàscula com el circuit electrònic. Sobretot és important preveure el cas de la pluja, mirant clot quedi "hermèticament" tancat, que no estigui en un lloc on s'hi formin bassals, etc.

5. CONCLUSIONS

Cal remarcar que el que s'ha fet és un estudi de viabilitat, és a dir, desenvolupar l'electrònica i el software base per comprovar que els diferents components eren compatibles entre sí. En el cas del microcontrolador, per tant, n'hem tingut prou amb un ATmega8 per comprovar i desenvolupar aquesta correcta interconnexió de tots els components. A partir d'aquí, escalar el sistema perquè permeti emmagatzemar dades durant mesos és només qüestió de canviar de microcontrolador i de fer petites adaptacions.

L'estudi m'ha permès veure quins components m'han anat bé i quins no. Així per exemple, he vist que la construcció amb una memòria SD, tot i que bastant avantatjosa pel que fa a prestacions, era massa complexa d'implementar. També s'ha observat que el sensor de pes escollit és massa precís per les prestacions que s'exigeixen, amb el que amb un sensor més barat haguérem reduït costos sense reduir funcionalitat.

En resum, vull destacar com a part positiva del projecte la seva multidisciplinarietat. S'ha treballat amb aspectes de tant baix nivell com cables, voltatges, i electrònica digital. Amb feines de nivell mig com la de la programació del microcontrolador (un llenguatge C bastant primitiu). Arribant a un últim nivell d'abstracció molt elevat: la programació en C#, usant orientació a objectes i gràfiques.

6. MILLORES

6.1 Escalat a microcontrolador més potent

Com hem dit, el microcontrolador que fem servir per aquest estudi (ATmega8) es queda curt en quant al nombre de pins per poder gestionar la memòria. Per escalar aquest sistema a un microcontrolador de la mateixa família però més potent (ATmega128) s'ha de començar adaptant el **circuit electrònic.** Només cal ampliar el nombre de línies d'adreça (usar els pins A12, A14, A133, A8, A9, A11 i A10 que actualment no s'usen). La resta de línies, les de control i les de dades, no cal tocar-les.

Aquest canvi comporta una revisió en el software del **microcontrolador** en el que respecta al nombre de dades a incorporar. Però no es tracta de cap canvi estructural, ja que els bits de control funcionaran de la mateixa manera.

En el software de **descàrrega** no s'involucren canvis. I en el programa d'**anàlisi**, s'haurà de revisar l'objecte *temps*, que passarà a portar també els dies, així com les variables *codis* i *llistaCompleta*.

La part on més s'haurà de treballar és en la del software del mòdul de memòria del microcontrolador, mentre que la resta el codi ja ha estat pensat per a ser fàcilment escalable.

6.2 Solució alternativa a l'escalat: Memòria SD

Una solució alternativa a la memòria RAM i a l'escalat a un microcontrolador més gran és l'ús d'una targeta SD (memòria flaix) per a emmagatzemar totes les dades. L'avantatge més destacat és la gran quantitat de memòria que es pot usar amb aquesta tecnologia. Per altra banda, el nombre de pins utilitzats és molt menor (4 pins), i no augmenta en funció de la quantitat de memòria que vulguem indexar.

Per contra, amb aquesta tecnologia és més difícil la depuració, en tractar-se d'un protocol SPI. Una mostra d'un sistema construït amb aquesta tecnologia es pot trobar a [8] i [9].

6.3 Incorporació de nous sensors

Una vegada tenim el sistema base, afegir nous sensors no implicaria grans canvis estructurals al sistema. Només *ampliacions*. Es podrien afegir varis tipus de sensors. Un sensor de llum com ara un fotodiode, que té més rang de *lux* que una fotoresistència. Aquest sensor es connectaria a una entrada ADC tal i com hem fet amb el sensor de temperatura. També seria interessant un sensor d'humitat i un sensor de pressió. De sensors d'aquests dos tipus n'hem trobat per a aquest projecte, amb l'impàs que són CI que usen SPI, un protocol que no tenim implementat, amb el qual aquí ja no es tractaria només d'un escalatge sinó de tota la codificació d'un mòdul de comunicació SPI.

Referències

[1] *Informació Strain Gauges* - http://techscienceonline.com/StrainG.aspx

[2] *Producte Futek* - http://www.futek.com/product.aspx?stock=FSH01455

[3] *Informació Load cells* - http://www.societyofrobots.com/sensors_forcetorque.shtml

[4] *Producte Scales-r.us* - http://www.scales-r.us/10kglc.htm

[5] *Producte de Robotshop* - http://www.robotshop.ca/interlink-05-circular-fsr.html

[6] *Bluemore 200* - www.eikonsite.it

[7] *PonyProg* - http://www.lancos.com/prog.html

[8] *ATmega amb targetes MMC* - http://www.captain.at/electronic-atmega-mmc.php

[9] *ATmega amb targetes SD* - http://www.captain.at/electronic-atmega-sd-card.php

RESUM

Aquest projecte tracta sobre la viabilitat de la construcció d'un sistema per al seguiment del pes d'una població de marmotes en alta muntanya. Bàsicament, es construeix una bàscula amb un sensor de força i un sensor de temperatura. Aquestes sortides analògiques es connecten a un microcontrolador ATmega8 que, mitjançant un algorisme desenvolupat en aquest projecte, està contínuament en escolta fins a detectar un canvi sobtat en el pes. Aleshores les dades s'enregistren i es guarden en una memòria SRAM per a, posteriorment, poder ser descarregades a un ordinador i analitzades per un programa que s'ha creat per a tal finalitat.

ABSTRACT

This project is about the construction feasibility of a system for the follow-up of a marmot's population weight in high mountain. We build a weighing scale with a force sensor and a temperature sensor. These analog outputs connect to an ATmega8 microcontroller that, with an algorithm developed in this project, is in continuous listening to detect sudden changes in the weight. When this happens, the data is recorded in a SRAM memory for, later, being able to download this data to a computer for it to be analyzed by a program that has been created for such purpose.

RESUMEN

Este proyecto trata sobre la viabilidad de la construcción de un sistema para el seguimiento del peso de una población de marmotas en alta montaña. Básicamente, se construye una báscula con un sensor de fuerza i un sensor de temperatura. Estas salidas analógicas se conectan a un microcontrolador ATmega8 que, mediante un algoritmo desarrollado en este proyecto, está en continua escucha hasta detectar un cambio repentino en el peso. Entonces los datos se registran y se guardan en una memoria SRAM para, posteriormente, poder ser descargadas a un ordenador y analizadas por un programa que ha sido creado para tal fin.