Universidad de Palermo

Facultad de Ingeniería

TRABAJO FINAL DE GRADO

MEDICIÓN HIDROLÓGICA DEL CAMPO PARA LA MEJORA DE PRODUCCIÓN

Profesor Tutor: Cristian Conejeros
Alumno: Naboni Mariano Pablo
Legajo: 0071480
E-mail: marianonaboni@gmail.com

CONTENIDOS

FIGURAS

TABLAS

1. Introducción

Desde los inicios del cultivo de la tierra, el conocimiento y entendimiento de las condiciones climáticas fueron factores determinantes para el desarrollo de la agricultura.

El incremento de la densidad poblacional mundial, genera la necesidad de un aumento en la producción de materias primas para la elaboración de alimentos. Obligando de esa manera a perfeccionar e industrializar la actividad agrícola a través del desarrollo de nuevos métodos y tecnologías.

En la actualidad, conocer y medir la precipitación en forma de lluvia, como así también obtener valores de temperatura y humedad en las zonas de cultivo, es un factor fundamental para lograr una eficiente producción agrícola.

A través de la implementación de instrumentos de medición y recolección de datos climáticos, hoy se permite a los productores tomar mejores decisiones y aprovechar de forma eficiente este bien escaso y necesario, para obtener el máximo provecho de su cultivo.

A lo largo de este trabajo, se mostrará el desarrollo, el armado y la aplicación de un sistema de medición y adquisición de datos inalámbricos, que obtiene y almacena valores de temperatura, humedad y precipitación caída en un punto específico.

El caso de estudio elegido es en la zona rural del partido bonaerense de Pergamino.

En la actualidad la recolección de datos de variables es deficiente y recae directamente sobre la producción. Por lo tanto en el siguiente proyecto trataremos de demostrar los beneficios de contar con una herramienta práctica y funcional para la obtención de datos que luego se reflejara y proyectara en beneficios a la hora de tomar decisiones para lograr un óptimo cultivo.

2. Definición de problemas

En el mundo el desafío en materia alimenticia se centra en la mejora en la producción agropecuaria, y son los factores climáticos los que más inciden en el resultado de la misma.

El aumento de la población mundial, genera una mayor necesidad de fuentes de alimento, esto desencadena una expansión de las tierras para la producción de materias primas de más de 6 millones de kilómetros cuadrados.

El siguiente grafico proporcionado por la ONU podemos observar el abrupto incremento en la cantidad de kilómetros cuadrados de tierra cultivable del año 1991 al 1992 en el mundo.

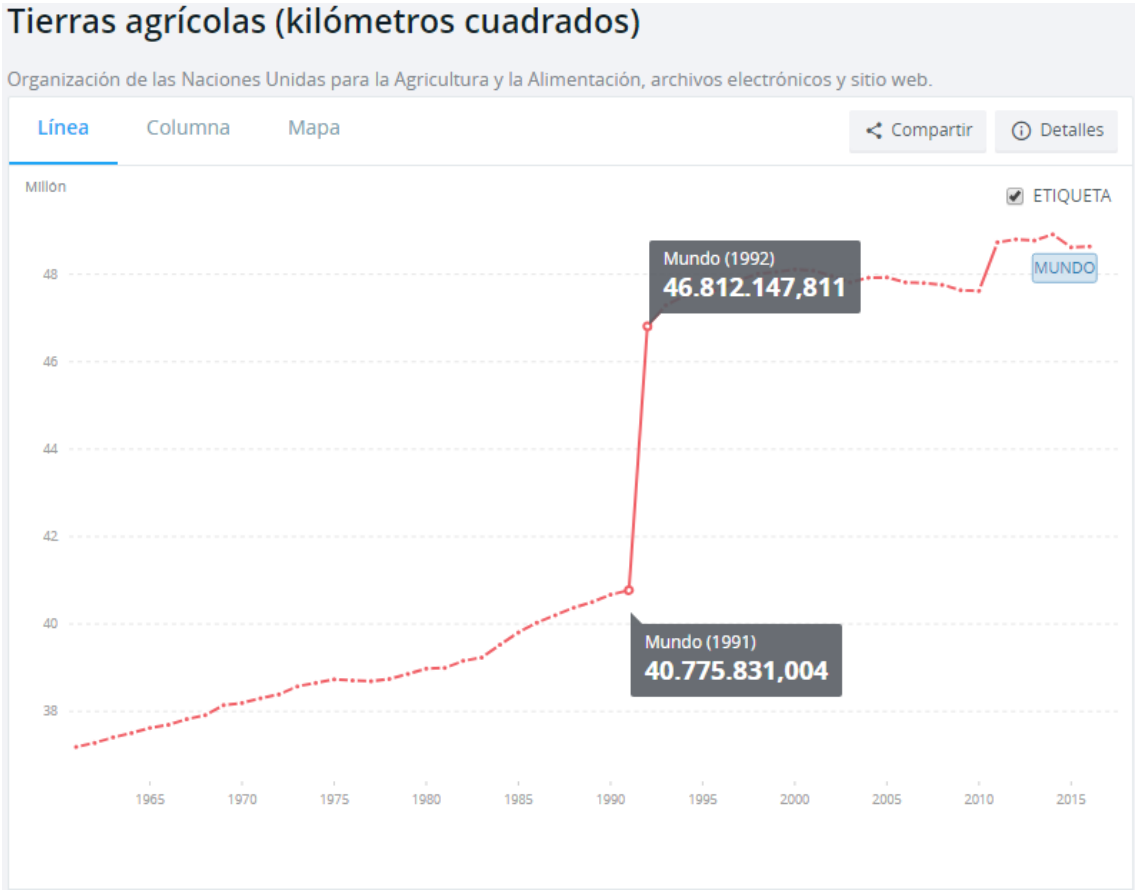


Figura 1 - Curva de crecimiento de tierras cultivables

El factor climático más determinante en la producción es la lluvia, ya que la escasez o abundancia de la misma afecta directamente en el rendimiento de la producción.

Hoy día, para obtener datos sobre las precipitaciones y poder actuar en consecuencia, se instalan centrales meteorológicas conectadas a internet, lo cual permite hacer un seguimiento online. Al momento, esto ocurre solamente en países desarrollados que cuentan con una infraestructura de redes y sistemas de comunicaciones que permiten dar soporte las plataformas de registro y análisis climático.

2.1.Ventajas y Riesgos

Conocer la cantidad de lluvia, humedad y temperatura en una determinada zona, es sumamente importante ya que permite:

- Actuar ante la espera de la siembra de un nuevo cultivo.
- Saber si es posible fumigar o no.
- Tomar decisiones sobre la cosecha.
- Determinar la calidad del cultivo a obtener.

Por otro lado, los riesgos de no contar con información respecto a la cantidad de lluvia, humedad y temperatura son:

- Posibilidad de perder la oportunidad del momento de siembra, perdiendo rinde.
- Pérdida del cultivo por exceso de lluvia
- Fumigación sin efecto por causa de lluvia.
- Posibilidad de dejar pasar la cosecha por errónea percepción de la humedad del suelo.
- Perdida de rinde por no actuar a tiempo por inclemencias del clima.

2.2.Métodos actuales utilizados en el país

En nuestro país las tecnologías disponibles para la adquisición de datos referidos a la lluvia son:

- Medición con centrales meteorológicas en la zona urbana
  Contra: se encuentra alejada del campo, no brinda información precisa de la zona de cultivo.
- Verificación por medio de diferentes páginas web como www.windguru.cz , www.accuweather.com o www.meteored.com.ar
  Contra: no brinda información precisa de la zona de cultivo. Dependencia de acceso a internet
- Medición con vaso pluviómetro graduado
  Contra: medición inexacta y requiere presencia en el lugar
- Centrales alimentadas con grupos electrógenos
  Contra: requiere presencia en el lugar para la descarga de datos.
- Sistema de mensajes vía SMS celular
  Contra: requiere soporte de PC para análisis de los datos

2.3.Problemática Puntual

El caso de estudio propuesto, se desarrolla en la zona de Pergamino, provincia de Buenos Aires.

Figura 2 - Área de estudio

De acuerdo a nuestra salida de campo, en la zona de estudio no hay actualmente en uso, un sistema para determinar o conocer en tiempo real, cuanto llovió, a que temperatura y en qué condiciones de humedad se encuentra en un lugar específico.

El relevamiento realizado con los vecinos de la zona, reflejo un creciente interés en la utilización de nuevas tecnologías que permitan acceder a historiales climáticos en puntos específicos.

El siguiente trabajo final de grado propone una solución a la problemática mencionada.

3. Objetivos

3.1. Objetivos Generales

Aportar una solución que permita conocer la cantidad de lluvia depositada en un lugar específico con características de registro y accesibilidad.

- REGISTRO:
  Generación de un registro histórico para analizar y sacar conclusiones.

- ACCESIBILIDAD:
  Posibilidad de ser visualizado desde cualquier lugar del mundo.

3.2. Objetivos Específicos

1. Entender los beneficios de la medición de la lluvia.
   1.1. Obtener mayor conocimiento de los sistemas de medición de lluvia
   1.2. Entender los principios de funcionamiento de las mediciones.

2. Diseñar y armar un sistema que permita medir la lluvia en un punto específico.

3. Diseñar una plataforma para la presentación de los resultados obtenidos de forma intuitiva y práctica, que permita el análisis de datos y de registros históricos.

4. Desarrollo.

Se propone la construcción de un sistema que reporte en cada intervalo de tiempo determinado, la lluvia caída en ese lapso, la temperatura y la humedad ambiente.

Características requeridas

- Independencia de la red eléctrica: ya que las distancias a los tendidos eléctricos suele ser de difícil acceso en la zona.
- Servicio de Reporte a un servidor: capaz de almacenar la información para futuro análisis.
- Sistema autónomo: que permita chequear constantemente el correcto funcionamiento de los dispositivos, actuar y reportar en caso de mal funcionamiento.

4.1. Tecnologías usadas

La medición de la lluvia data de la antigua Grecia, 500 años a.c. y desde aquella época, la cantidad de lluvia para los cultivos tiene relevancia para el cultivo.

El principio básico utilizado es un recipiente graduado, el cual indica la altura de lluvia depositada en una superficie.

El recipiente graduado tiene un embudo para mejorar la captación de lluvia.

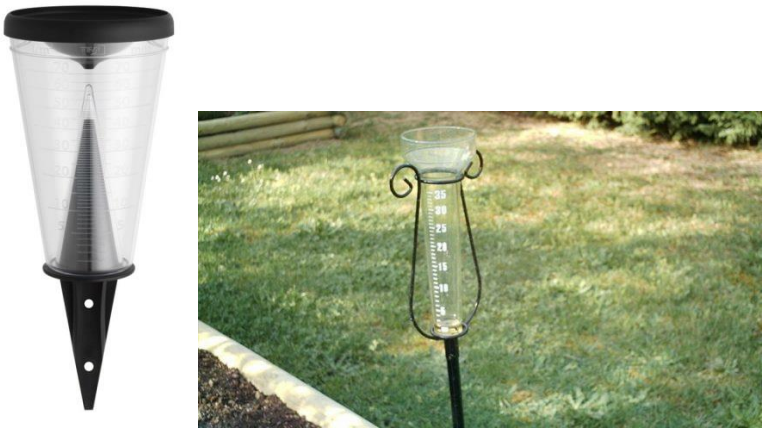


Figura 3 - Pluviómetros de uso común.

Hay formas y tamaños muy variados, pero todos bajo el mismo principio, apreciación visual de un medidor la cantidad de lluvia. La unidad de medida es: altura en milímetros / metros cuadrados.

4.2.Adquisición de datos en el caso de estudio

Entre los meses de febrero y abril del año 2018, se realizó un relevamiento de la zona rural del partido de Pergamino.

Dicho estudio arrojo los siguientes métodos de estimación de precipitación caída

- Pluviómetro del tipo medición manual
- Comunicación telefónica entre campos vecinos para comunicarse las mediciones por zonas.
- Estaciones meteorológicas para ver datos online en la zona urbana.
- Estaciones meteorológicas alimentadas por baterías en la zona de trabajo que requieren presencia en el lugar para bajar la información en una pc y posteriormente hacer un análisis de lo ocurrido.

Ante este escenario, se propone el desarrollo de un sistema de medición y adquisición con capacidad de reporte a una plataforma basada en servidor.

Este sistema permitirá calcular la lluvia caída en un periodo de tiempo establecido, la temperatura y la humedad ambiente.

Es un modo efectivo de obtención de información al instante, que brinda un servicio de respuesta rápida, precisa y útil.

4.3.Recursos disponibles en el caso de estudio

Otro dato de interés que se desprendió del relevamiento, fueron los recursos disponibles en el área de estudio.

Electricidad	Estado	Comunicación	Estado
Cable (220V)	No	Cableado	No
Estación energía(12V)	No	Redes Lora	No
Vientos	Mínimos	Redes Sigfox	No
Luz solar	Alto	Redes GPRS	Si

Tabla 1 - Recursos disponibles.

Este análisis sirvió de soporte para poder comprender más sobre el lugar de trabajo de y trabajar en la búsqueda de la solución.

4.4.Descripción del sistema

El sistema propuesto consta de las siguientes partes: micro controlador, módulos de medición, sistema de comunicación, sistema de abastecimiento de energía y servidor.

- Micro controlador: es el encargado de mantener todo los módulos de medición funcionando. Se ocupa tanto de leer, como de recibir los datos de los dispositivos.
- Módulos de medición: son el pluviómetro, un sensor de temperatura y humedad.
- Sistema de comunicación: es el encargado de trasmitir los datos a la nube para su posterior análisis.
- Sistema de energía: es el encargado de alimentar eléctricamente al sistema
- Servidor: es el encargado de interpretar, analizar, guardar y mostrar los datos que recibe.

5. Metodología general

El equipo desarrollado funciona a modo de estación meteorológica,

- Permite el acceso y la visualización de datos en forma remota desde cualquier lugar del mundo.
- Envía datos a la nube para su utilización en el momento y análisis posterior.
- Usa un sistema de pívot para la medición de lluvia.
- Implementa un sensor DHT22 para medición de humedad y temperatura.
- Es un sistema autónomo que requiere luz solar y acceso a internet para su funcionamiento a través de las redes GPRS.

6. Fundamentación

Si bien existen diversas fuentes de información para constatar la cantidad de precipitación sobre una zona, como así también páginas del clima o pronósticos del tiempo, la realidad es que no se obtienen valores exactos sobre la cantidad de lluvia caída en alguna parte determinada de interés. Esto básicamente se debe a que los sensores se encuentran distantes de la zona de trabajo.

Sabiendo de antemano la cantidad de lluvia, se puede programar la pulverización de agroquímicos, calcular momentos de siembra o definir una cosecha dependiente de la humedad.

A su vez, en los casos donde no se cuente con las lluvias necesarias, se puede implementar una mejor utilización de los sistemas de riego.

En los tiempos modernos que corren es importante no perder tiempo y combustible en realizar una verificación en un lugar remoto, este nuevo método se realiza en la nube, optimizando el tiempo del usuario y la toma de decisiones.

Uno de los inconvenientes a afrontar son, los escasos o nulos recursos de tendidos eléctricos.

La falta de medios de comunicación inalámbricos, llevaron a la utilización redes inalámbricas 2G/3G que fue la existente en el lugar.

7. Descripción general del equipo

7.1.Componentes

Descripción del sistema, sus componentes:

Gabinete: Caja estanca de 162x212x110mm.

Sensores: Pluviómetro de estación meteorológica Meteo Star WH 1080/81
Sensor de humedad y temperatura DHT22.
Fuente de energía: Panel solar de 10W
Almacenamiento: Batería Maura de 12v 7Ah.
Control de carga: Regulador de carga SMART 12/24v.

Plataforma de procesamiento: Placa de diseño propio con un micro-controlador Atmega128.

Sistema de comunicación: GPRS con placa de la marca RoHs.

7.2.Diagrama del sistema

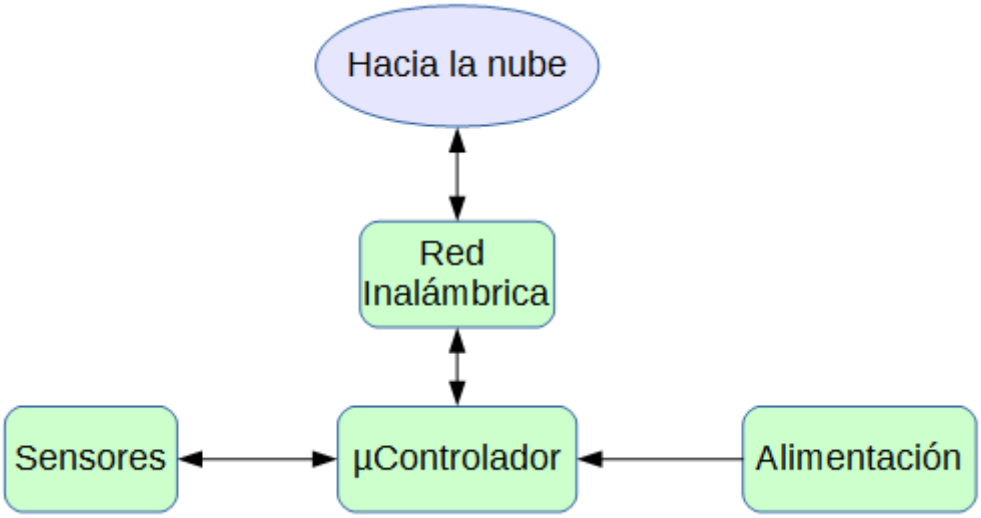


Figura 4 - Diagrama en bloques del sistema.

El diagrama es un esquema de lo realizado, una estación meteorológica autónoma, que se relaciona con diferentes sensores, envía la información por un sistema inalámbrico comunicándose con un servidor.

Gracias a la información guardada se puede hacer un análisis de los datos, permitiendo poder ver en tiempo real lo que está ocurriendo desde cualquier punto del mundo en cualquier dispositivo móvil.

7.3.Diagrama de flujo

Todo el sistema tiene una función de recuperación de estados, tanto control de errores como perdida se conexión.

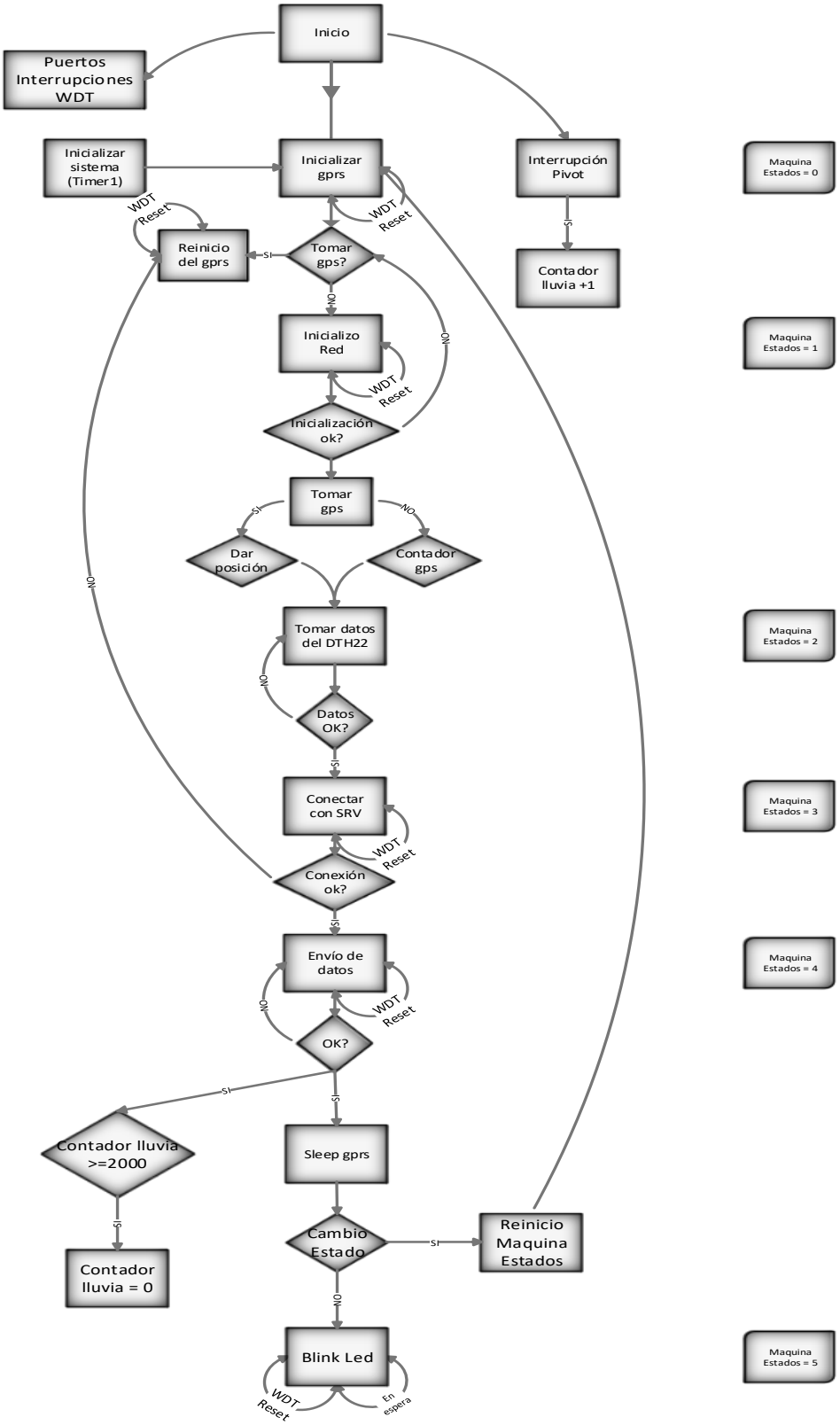


Figura 5 - Diagrama de flujo

8. Descripción particular del equipo

Descripción de cada parte del equipo, su forma de medición y su configuración.

8.1. Construcción y diseño de la placa de desarrollo

Consiste en el armado de una placa lo suficientemente versátil que permita la escalabilidad hacia otros sensores o dispositivos.

8.1.1. Circuito implementado

Placa desarrollada con el software EAGLE versión 8.7.1

Diseño pensando en la opción de futuros puertos a utilizar.

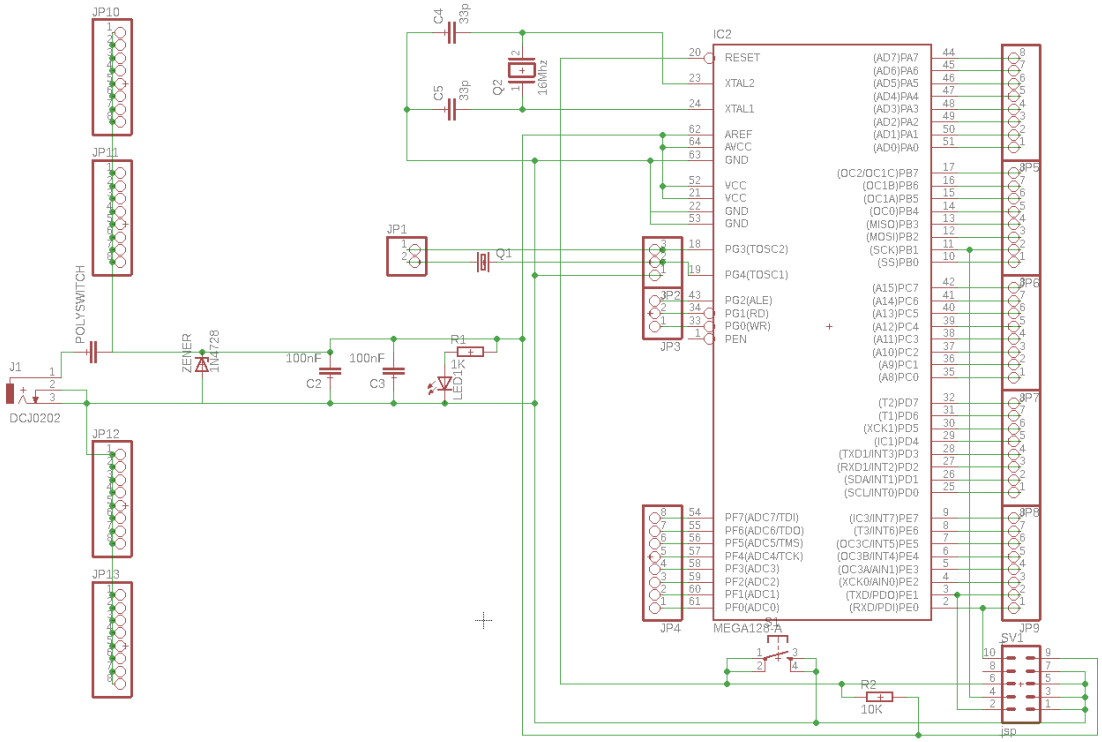


Figura 6 - Diagrama de placa con Atmega128

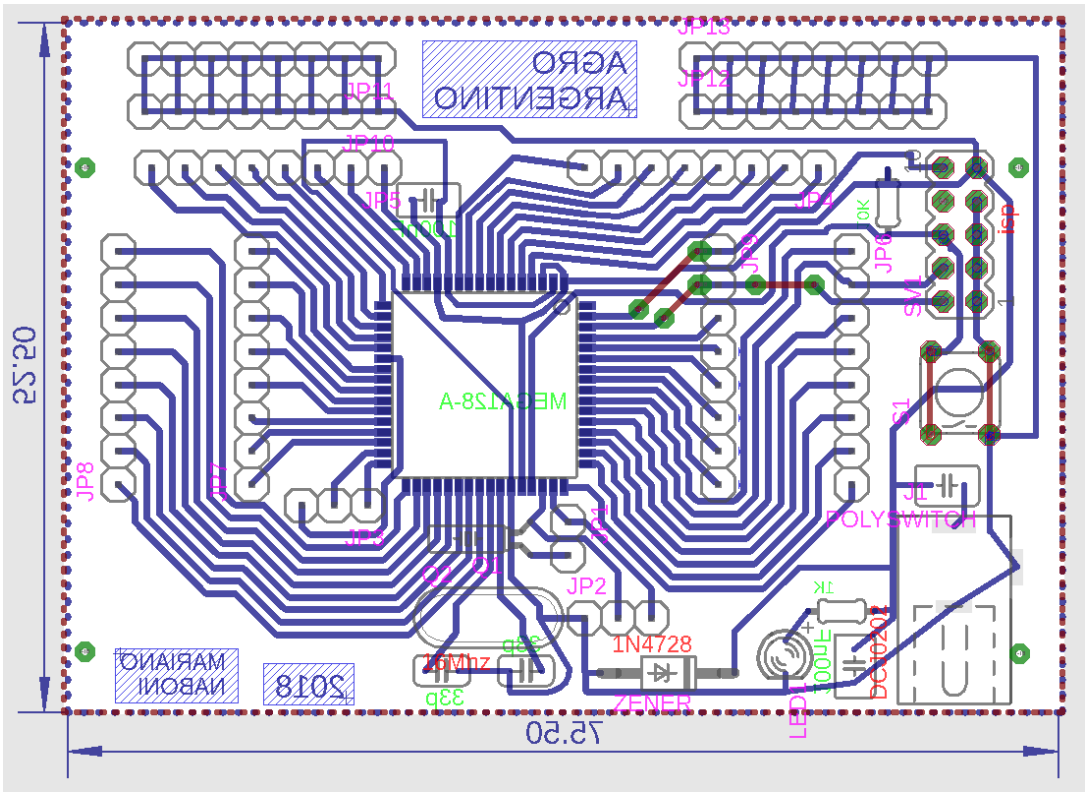


Figura 7 - Componentes y pistas

Pines adiciónales para energizar más dispositivos y lograr un mejor aprovechamiento de los puertos disponibles.

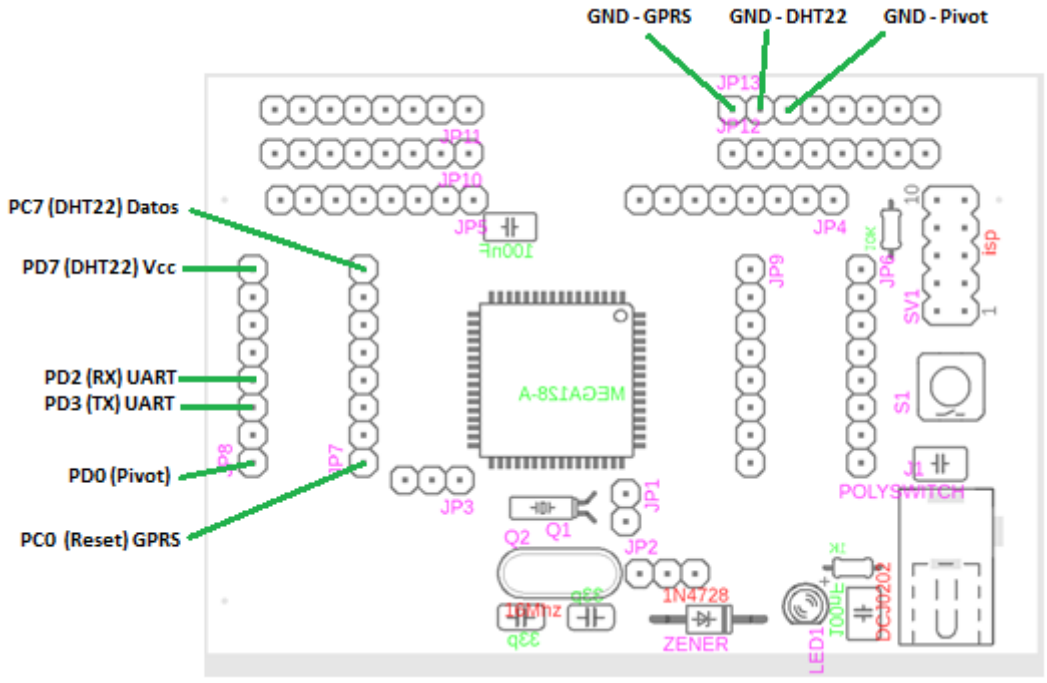


Figura 8 - Componentes y conexiones de los módulos.

8.2.Micro controlador

El trabajo fue realizado con un micro-controlador de la familia Atmega. Modelo: ATMEGA128

El mismo fue elegido debido a sus principales características:

- Fácil adquisición en el mercado
- Económico.
- Buena capacidad de procesamiento.
- Posibilidad de escalabilidad del proyecto
- Ampliamente conocido e información técnica accesible
  .

Figura 9 - Micro controlador Atmega128

8.2.1. Características principales y puertos de salida/entrada

En la siguiente imagen se aprecia:

- Diagrama de pines de atmega128
- Distribución de puertos

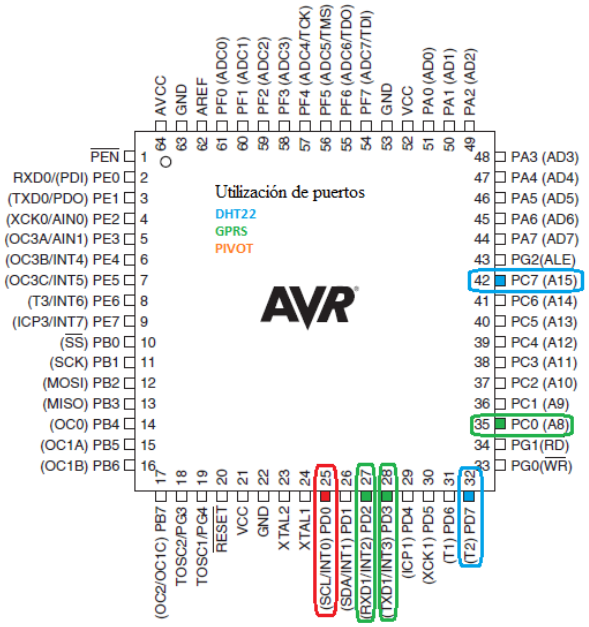


Figura 10 - Puertos entrada/salida Atmega128

Características utilizadas más importantes:

- UART
- Timer de 16 bits
- Watchdog
- Interrupciones externas configurables Int0.

Para mayor detalle técnico del respectivo microcontrolador, ver anexo 2 “Datasheet ATMEGA128”.

9. Configuraciones del sistema

9.1.Programación

Velocidad del micro controlador.

Como criterio de diseño se implementa una velocidad baja de 1Mhz utilizando el oscilador interno, a menor frecuencia menor consumo.

Se deja preparado el sistema para poder colocar un oscilador de cristal externo si es necesario darle mayor precisión, los cambios de temperatura pueden producir una variación en el oscilador interno.

Para la velocidad de trabajo se tiene que programar el micro controlador a través del puerto SPI del mismo, el USBAsp, es un programador para micro controladores Atmega a través del puerto SPI.

La configuración de fusibles internos son los que le indican al micro controlador como va a funcionar, la página http://www.engbedded.com/fusecalc/especifica las configuraciones de los fuses (fusibles) del micro controlador.

Seleccionado el tipo de microcontrolador, se indica a qué velocidad va a funcionar.

La opción de compatibilidad Atmega103, es para tener habilitadas configuraciones especiales, para tener todas las opciones del atmega128, no se aplica.

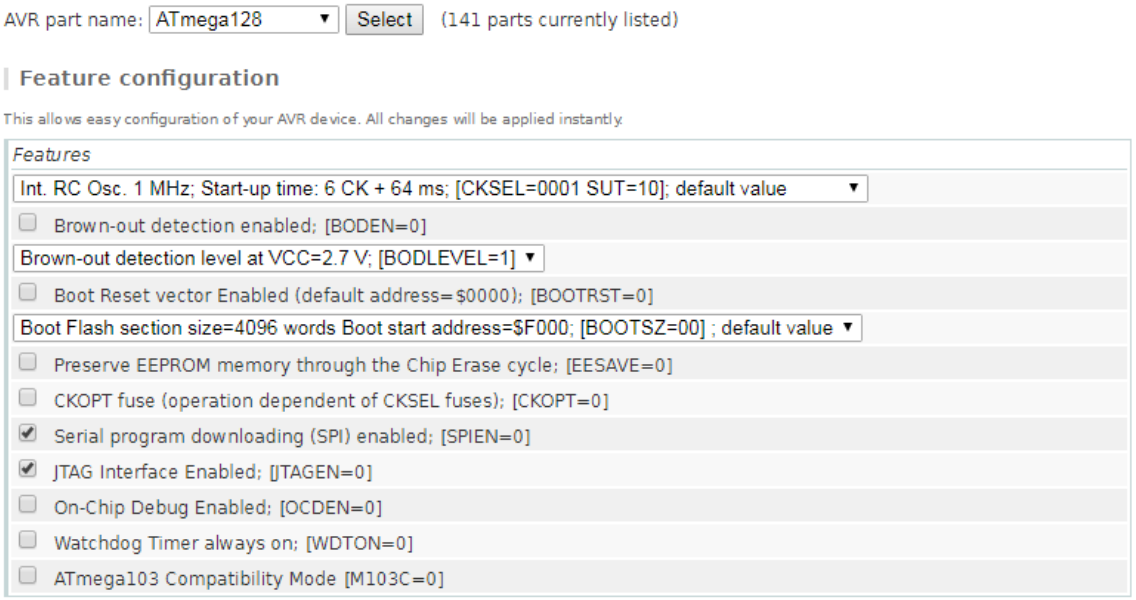
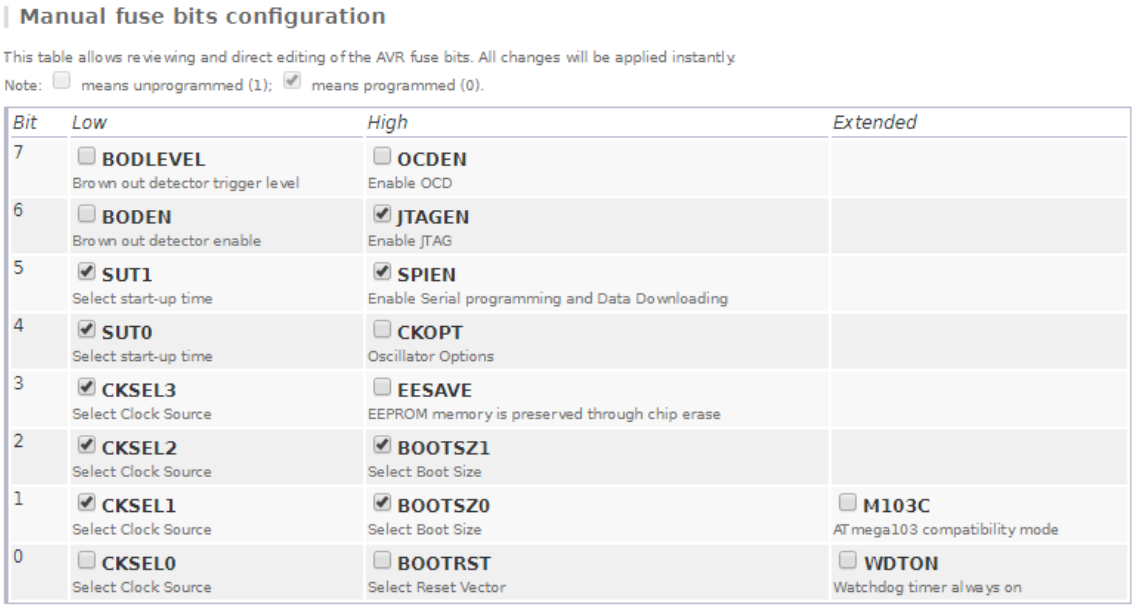


Figura 11 - Configuración de fusibles para programar el microcontrolador.

El retardo de 64ms es un espera en el inicio para que los demás dispositivos se inicialicen antes de que inicialice el sistema, en este caso no es necesario, pero como practica de implementación, se considera mejor.

La configuración manual de los bits de los fusibles queda visualizada de la siguiente manera:



Manual fuse bits configuration

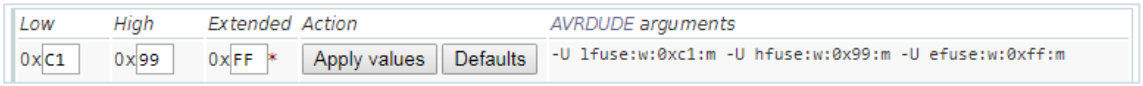
This table allows reviewing and direct editing of the AVR fuse bits. All changes will be applied instantly.

Note: ☐ means unprogrammed (1); ☑ means programmed (0).

Bit	Low	High	Extended
7	☐ **BODLEVEL** Brown out detector trigger level	☐ **OCDEN** Enable OCD	
6	☐ **BODEN** Brown out detector enable	☑ **JTAGEN** Enable JTAG	
5	☑ **SUT1** Select start-up time	☑ **SPIEN** Enable Serial programming and Data Downloading	
4	☑ **SUT0** Select start-up time	☐ **CKOPT** Oscillator Options	
3	☑ **CKSEL3** Select Clock Source	☐ **EESAVE** EEPROM memory is preserved through chip erase	
2	☑ **CKSEL2** Select Clock Source	☑ **BOOTSZ1** Select Boot Size	
1	☑ **CKSEL1** Select Clock Source	☑ **BOOTSZ0** Select Boot Size	☐ **M103C** ATmega103 compatibility mode
0	☐ **CKSEL0** Select Clock Source	☐ **BOOTRST** Select Reset Vector	☐ **WDTON** Watchdog timer always on

Figura 12 - Configuración de opciones de fusibles del Atmega128.

Esta configuración devuelve una línea de comandos para poder programar el micro controlador a la velocidad especificada.



Low	High	Extended	Action	AVRDUDE arguments
0xC1	0x99	0xFF *	Apply values Defaults	-U lfuse:w:0xc1:m -U hfuse:w:0x99:m -U efuse:w:0xff:m

Figura 13 - Líneas de comandos para programar Atmega128

Desde la consola CMD de Windows se utilizan los siguientes comandos:

```
c:\WinAVR-20100110\bin>avrdude.exe -c usbasp -p m128 -U lfuse:w:0xc1:m -U
hfuse:w:0x99:m -U efuse:w:0xff:m
```

Donde, **usbasp** es el tipo de programador, **m128** es el micro controlador a programar y el resto son las configuraciones establecidas anteriormente.

Desde la consola de Windows la respuesta a los comandos es la siguiente:

```
C:\WinAVR-20100110\bin>avrdude.exe avrdude.exe -c usbasp -p m128 -U lfuse:w:0xc1
:m -U hfuse:w:0x99:m -U efuse:w:0xff:m

avrdude.exe: warning: cannot set sck period. please check for usbasp firmware up
date.
avrdude.exe: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.03s

avrdude.exe: Device signature = 0x1e9702
avrdude.exe: reading input file "0xc1"
avrdude.exe: writing lfuse (1 bytes):

Writing | ################################################## | 100% 0.02s

avrdude.exe: 1 bytes of lfuse written
avrdude.exe: verifying lfuse memory against 0xc1:
avrdude.exe: load data lfuse data from input file 0xc1:
avrdude.exe: input file 0xc1 contains 1 bytes
avrdude.exe: reading on-chip lfuse data:

Reading | ################################################## | 100% 0.02s

avrdude.exe: verifying ...
avrdude.exe: 1 bytes of lfuse verified
avrdude.exe: reading input file "0x99"
avrdude.exe: writing hfuse (1 bytes):

Writing | ################################################## | 100% 0.02s

avrdude.exe: 1 bytes of hfuse written
avrdude.exe: verifying hfuse memory against 0x99:
avrdude.exe: load data hfuse data from input file 0x99:
avrdude.exe: input file 0x99 contains 1 bytes
avrdude.exe: reading on-chip hfuse data:

Reading | ################################################## | 100% 0.02s

avrdude.exe: verifying ...
avrdude.exe: 1 bytes of hfuse verified
avrdude.exe: reading input file "0xff"
avrdude.exe: writing efuse (1 bytes):

Writing | ################################################## | 100% 0.02s

avrdude.exe: 1 bytes of efuse written
avrdude.exe: verifying efuse memory against 0xff:
avrdude.exe: load data efuse data from input file 0xff:
avrdude.exe: input file 0xff contains 1 bytes
avrdude.exe: reading on-chip efuse data:

Reading | ################################################## | 100% 0.01s

avrdude.exe: verifying ...
avrdude.exe: 1 bytes of efuse verified

avrdude.exe: safemode: Fuses OK

avrdude.exe done.  Thank you.
```

Figura 14 - Salida de consola de programación Atmega128

Al final, se hace una verificación, dando el ok, significando que salió bien la configuración aplicada.

9.2.Interrupciones

Hay cuatro interrupciones las cuales se detallan de la siguiente manera:

1) Watchdog, para recupero del sistema en caso de falla
2) Int0, interrupción del puerto PD0,
3) Timer1, interrupción producida por un contador,
4) UART2, interrupción por ingreso de datos por UART.

9.2.1. Watchdog (WDT)

El sistema es suficientemente robusto para que pueda reaccionar ante una falla.

Si tiene una demora en la respuesta de cualquier dispositivo, esta función reiniciara el equipo.

La función tiene un contador de tiempo de vida, si no se reinicia el contador de tiempo de vida antes del tiempo establecido se reiniciara todo el sistema.

En el inicio del programa se comienza con la función de WDT apagada, debido a que si se inicia y tarda más de lo establecido, todo el sistema estaría reiniciándose, y nunca arrancaría completamente el sistema.

Las librerías utilizadas son #include avr/wdt.h.

Se utiliza wdt_disable (); al comienzo del inicio para tenerlo apagado.

Para el tiempo de iniciación es utilizada la función wdt_enable (WDTO_2S), de donde lo que está entre paréntesis son tiempos preestablecidos para configurar.

Las opciones de configuración son:

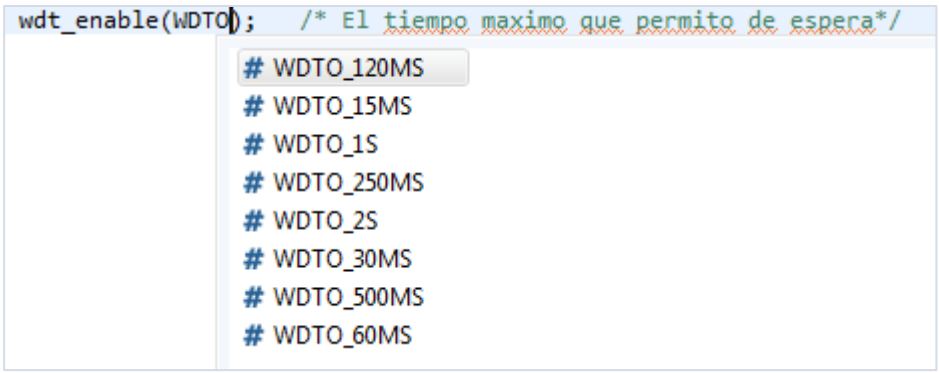


Figura 15 - Opciones de configuración de tiempos del Watchdog.

Las diferentes opciones a configurar, que ya viene armada en la librería, son cambiando los valores del control de registro del watchdog (WDTCR).

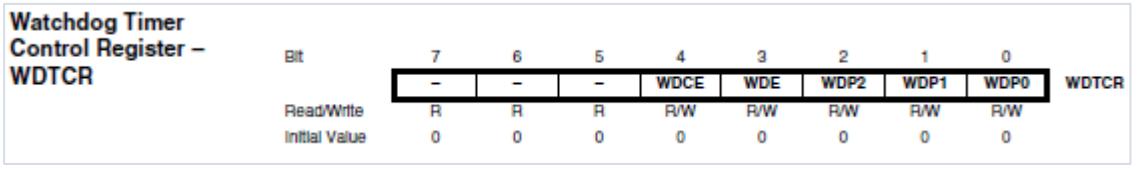


Figura 16 - Control del vector de registros del temporizador del Watchdog.

La configuración de los flags WDP(X) son los encargados de configurar el tiempo que se necesite antes de reiniciar el sistema en caso de fallo.

WDP2	WDP1	WDP0	Tiempo de espera típico en VCC = 5.0V
0	0	0	16.3ms
0	0	1	32.5ms
0	1	0	65ms
0	1	1	0.13s
1	0	0	0.26s
1	0	1	0.52s
1	1	0	1.0s
1	1	1	2.1s

Figura 17- Opciones del registro del temporizador del Watchdog.

Configurado el tiempo de trabajo, se reiniciara el contador del Watchdog para que no llegue su contador al tiempo establecido y produzca un reinicio del sistema.

La función wdt_reset (); pone en cero el contador del Watchdog.

9.2.2. INT0

El sistema de pívot da un cambio de estado alto ha estado bajo, cuando sucede esto, se activa la interrupción y se inicia un contador, este dará la cantidad de veces que el pívot cambia de estado, sabiendo cuantos mm representan cada cambio, se puede sacar la cantidad de mm llovidos.

Para captar este cambio se detecta el cambio de estado del puerto.

Es necesario que se configure el puerto de la siguiente forma según la información brindada por la hoja de datos.

Para indicar como se va a activar la interrupción externa, puede ser por nivel o por flanco, tengo que configurar EICRA, que va desde la interrupción 3 a la 0.

Bit	7	6	5	4	3	2	1	0	
	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 18 - Registró EICR para la Interrupción.

Se deja configurado la interrupción ISC00, que me indica la interrupción 0 en el puerto PD0 por flanco ascendente.

Para activar la interrupción se utiliza el registro EIMASK.

Bit	7	6	5	4	3	2	1	0	
	INT7	INT6	INT5	INT4	INT3	INT2	INT1	IINT0	EIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 19 - Registró EIMASK.

Se pone en estado alto a INT0 para dejarlo activado.

La función que configurara la interrupción que será utilizada para el pluviómetro de pivote es:

```
void CONT_Init(){
/*Inicialización de la configuración de la interrupción INT0 */
      EICRA |=(1<<ISC00);     /* Configuración de INT0 */
      EIMSK |=(1<<INT0);      /* Activo la interrupción INT0*/
}
```

Como el cambio de transición suele tardar, se genera un mínimo retardo, el cual es mayor que el cambio necesario para detectar si fue verdadero o no la interrupción.

```
/**********Interrupción para el medidor de agua************/
/***********************Contador ***************************/
ISR(INT0_vect){/*PD0-->>Por donde activo la interrupción externa */
      if(TestBit(PIND,0)){    /*Si está activo tengo un evento*/
                  Pulsado=1;  /*Si estoy en la interrupción*/
      }
      else{
                  Pulsado=0;  /*Si entre en la interrupción pero no alcance
                              a leer el cambio*/
      }
      val1=Pulsado;     /*Leo el primer estado del pulsador*/
      _delay_us(10);
      if(val1!=botonestado){ /*El estado del botón ha cambiado*/
            if(val1==1){      /*Compruebo si el botón sigue apretado*/
            _delay_us(20);
            if(modo==0){            /*Sigo en estado alto?*/
                  modo=1;           /*Cambio de modo*/
                  contadorPivot=contadorPivot+1;
                        /*Entro en la interrupción sumo uno*/
                  }
            else{
                  modo=0;
```

```
                    contadorPivot=contadorPivot+1;
                          /*Entro en la interrupción sumo uno*/
                    }
              }
        }
        botonestado=val1; /*guardo el nuevo estado del botón*/
}
```

9.2.3. Interrupción del timer1

La configuración del Timer1 es utilizada para iniciar la máquina de estados.

Este es un contador de 16bits, el cual, se implementa para entrar en la interrupción cada 4s, un número que resulto muy práctico, ya que es un múltiplo de los segundos de un minuto, según la velocidad a la cual corre el micro controlador, es la velocidad a la cual corre ese Timer.

Los modos principales que tienen los contadores son:

Modo Normal: El timer cuenta desde 0 a 2^8-1 = 255 en un contador de 8bits o de 0 a 2^16-1 = 65535 en caso de un timer de 16bits y se desborda reiniciando la cuenta. Se puede generar interrupción al desbordarse o cuando la comparación del conteo llegue a un valor determinado.

Modo CTC: En este modo el timer1 es reiniciado a 0 cuando se coloca un valor de comparación con el timer, cuando llega a este valor se produce un reinicio a cero del contador.

Modo Fast PWM: Este modo logra una onda PWM. Con cada cuenta el valor del timer utilizado se compara con un valor, cuando coinciden cambia el estado alto (HIGH) de los pines de salida PWM, y cuando se reinicia el timer este pin vuelve a cambiar su estado bajo (LOW), lo que nos da dos valores, estado alto y bajo de la salida del puerto, generando la PWM.

Modo Phase Correct PWM: Este modo es una onda PWM con mejor resolución en comparación del modo Fast PWM. El timer cuenta hacia adelante y luego hacia atrás antes de realizar el cambio de estado del pin PWM, es decir, si estamos en un timer de 8bits, este cuenta de 0 a 255 y al llegar a 255 cuenta de 255 a 0, logrando una salida PWM mejor definida pero de menor frecuencia.

9.3.Programación del timer1

La frecuencia de trabajo del micro controlador es de 1MHz.

Es utilizado el contador en modo CTC, cada vez que se realiza un desborde, el contador vuelve a cero, evitando que se produzca algún problema en el cálculo.

Los siguientes cálculos son realizados para dar con el tiempo que se configure para que se produzca la interrupción.

El prescaler reduce la frecuencia de trabajo y lograr el tiempo requerido.

Es configurado para nuestro propósito un prescaler de 1024, trabajando con la menor frecuencia posible, dando una frecuencia de trabajo de:

$$\frac{1000000}{64} = 15625Hz$$

Este número da la frecuencia del contador, en tiempo significa:

$$\frac{1}{15625Hz} = 0{,}000064s = 0{,}064ms$$

Cada 0,064ms cuenta el contador sumando uno en uno.

El desbordamiento que se quiere hacer es el número que le ponga al OCR1A, que es el número en el cual el contador volverá a cero, entonces para establecer una interrupción cada 4 segundos:

$$\frac{1000}{62500 * 0{,}064} = \frac{1}{4} = 0{,}25Hz$$

Esto significa que dada esa frecuencia, el tiempo que entra en la interrupción es:

$$T = \frac{1}{F} = \frac{1}{0{,}25Hz} = 4s$$

Si multiplica por 15, da 60 segundos, ya se tiene la base de 1minuto, a partir de acá, se puede dar el tiempo que necesite implementar en el sistema simplemente contando la cantidad de veces que entro en la interrupción.

9.3.1. Registros de configuración

Una breve explicación de cómo se deben configurar el timer 1.

TIMSK

Timer/Counter Interrupt Mask Register – TIMSK

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 20 - Registró TIMSK

La máscara de registro de la interrupción del contador es utilizada para habilitar la interrupción de los temporizadores.

Los bits de configuración del timer1 son los bits del 5 al 2 del TIMSK.

El bit OCIE1A para habilitar el modo CTC.

OCR1A es el número hasta el cual necesita llegar a comparar con el contador, ya que estamos en el modo CTC.

TCNT1 es desde cuándo empieza a contar.

TCCR1B es la configuración del prescaler.

Timer/Counter1 Control Register B – TCCR1B

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 21 - Registró TCCR1B

La siguiente tabla indica los tipos de prescaler configurables.

Table 62. Clock Select Bit Description

CSn2	CSn1	CSn0	Description
0	0	0	No clock source. (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/1 (No prescaling
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on Tn pin. Clock on falling edge
1	1	1	External clock source on Tn pin. Clock on rising edge

Figura 22 - Selección de prescaler.

La configuración del timer elegido es la siguiente:

```
voidtimer1_init()
{
/* El prescaler = 64 y CTC mode */
    TCCR1B |= (1 << WGM12)|(0<<CS12)|(1<< CS11)|(1 << CS10);
/* Inicialización del timer*/
    TCNT1 = 0;
/* Inicialización de numero de comparación*/
/* Se produce una interrupción cada 4 segundos*/
    OCR1A = 62500;
/* Inicialización de la comparación de variable*/
    TIMSK |= (1 << OCIE1A);
}
```

9.3.2. Interrupción con el Timer1 y configuración de la interrupción

Dentro de la interrupción la forma de cálculos de tiempo son los siguientes:

```
/******Interrupción de cada cuanto quiero que se inicie la máquina de estados*******/
ISR(TIMER1_COMPA_vect){
      interrupcion++;   /*4Segx15=1Minuto     (por     la     configuración realizada)*/
      if ( interrupcion==15*Minutos ) {
      /*Cada este tiempo inicializo la máquina de estados*/
            SaltoGPS++;
            if(SaltoGPS==120){ /*Reasignar la localización una vez al día*/
                  SaltoGPS=0;
            }
            interrupcion = 0;
            ESTADO_GPRS=0;
      }
}
```

Si se multiplica por la cantidad de minutos la máquina de estados, fácilmente se calcula el tiempo de envío de datos al servidor, en este caso se define minutos como la variable a tener en cuenta para el tiempo a establecer para la máquina de estados.

9.4.UART

Es configurada la UART1 debido a que la UART0 traía complicaciones con el programador, por otro lado, la configuración sobre la línea de puertos más utilizados quedando más prolijo y compacto.

Es utiliza la UART l por interrupción, debido a que los datos son esperados una vez enviado el comando AT, que es el utilizado en el módulo conectado a este puerto.

Para que no quede una función bloqueante, se tiene una espera de 2000 ciclos, en caso de no recibir datos, devuelve un error.

Parámetros para envío y recepción de la UART:

```
void UART_Init(unsignedlong BAUDRATE)
```

```
/* Inicializo los valores de la UART */
{
      UCSR1B |= (1 << RXEN1) | (1 << TXEN1) | (1 << RXCIE1);
      /* Habilito  envío y recepción */
```

```
	UCSR1C |= (1 << UCSZ10) | (1 << UCSZ11);
	/* Le pongo 8 bit datos y 1 bit de stop  */
	UBRR1L = BAUD_PRESCALE;
	UBRR1H = (BAUD_PRESCALE >> 8);
}
char UART_RxChar()		/* Dato char de recepción */
{
	while (!(UCSR1A & (1 << RXC1)));	/* Espero el dato recibido */
	return(UDR1);                                                    /*
Devuelvo lo que recibí */
}
void UART_TxChar(char data)	/* Dato char de transmisión */
{
	UDR1 = data;
	/* Escribo el dato a transmitir en UDR1 */
	while (!(UCSR1A & (1<<UDRE1)));	/* Espero hasta que se transmita
                                        y que el buffer este vacío.*/
}
voidUART_TxString(char *str)		/* Envío una cadena */
{
	int i=0;
	while (str[i]!=0)
	{
		UART_TxChar(str[i]);
		/* Envío hasta que este el buffer vacío  */
		i++;
	}
}
```

Para calcular la velocidad de la UART se utilizó la siguiente ecuación en la uart.h

```
#define BAUD_PRESCALE((((F_CPU/16)+(BAUDRATE/2))/(BAUDRATE))-1)
/* Prescaler que nos dará la velocidad que le configuremos en la main*/
```

Baud Rate (bps)	f_{osc} = 1.0000MHz			
	U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error
2400	25	0.2%	51	0.2%
4800	12	0.2%	25	0.2%
9600	6	-7.0%	12	0.2%
14.4k	3	8.5%	8	-3.5%
19.2k	2	8.5%	6	-7.0%
28.8k	1	8.5%	3	8.5%
38.4k	1	-18.6%	2	8.5%
57.6k	0	8.5%	1	8.5%
76.8k	–	–	1	-18.6%
115.2k	–	–	0	8.5%
230.4k	–	–	–	–
250k	–	–	–	–
Max (1)	62.5Kbps		125Kbps	

Figura 23 - Configuración de Baud Rate para configurar la UART.

F_CPU = Frecuencia del micro-controlador = 1MHZ.

BAUDRATE = Velocidad que quiero que trasmita = 2400 bauds.

$$\left(\frac{\left(\frac{1000000}{16}\right)+\left(\frac{2400}{2}\right)}{2400}\right)-1=25{,}54$$

BAUD_PRESCALE = 25→ Valor que queda configurado por truncamiento.

9.4.1. Interrupción de la UART

Como es utilizada la uart por interrupción, cada vez que es recibido un dato por uart, este se guardado en un vector.

```
ISR (USART1_RX_vect)
{
      MYSTRING_RESPUESTA[Counter] = UDR1; //Guardo mi respuesta
      Counter++;  //Aumento el contador (tamaño del buffer)
      if(Counter == TAM_BUFFER)     //Si llegue al max del tamaño
      Counter = 0;      //Voy al inicio del buffer
}
```

El tamaño del vector, tiene que ser el tamaño máximo del dato que puede llegar a ingresar.

Cada vez que se obtiene un dato correcto, el contador vuelve a cero, logrando con esto, que en la próxima petición de datos, los valores anteriores se pisen.

9.4.2. Ubicación GPS por GPRS

Para disminuir módulos conectados y un menor uso de energía, su utiliza la geolocalización por el GPRS.

El tamaño del string de datos, con el que se recuperan las respuestas, debe ser lo suficientemente grande como para almacenar el dato recibido.

Donde la variable de almacenamiento MYSTRING_RESPUESTA [Counter] tiene la longitud Counter que va aumentando hasta un máximo de TAM_BUFFER = 300.

La respuesta recibida es:

+CIPGSMLOC: 0,-58.394344,-34.588696,2018/11/14,00:12:38

La función "strncpy" es la encargada de obtener el valor que nos interesa.

```
strncpy(GPS, MYSTRING_RESPUESTA+14, 34);
```

- GPS es la variable tipo string en donde se almacena el dato que me interesa enviar.
- MYSTRING_RESPUESTA + 14, es a partir de qué valor dentro de la cadena de caracteres empiezo a guardar.
- 34, es cuantos caracteres guardo en la variable GPS.

Dando como resultado GPS=-58.394344,-34.588696,2018/11/14,

9.5.Puerto DHT22

El puerto utilizado, es configurado como escritura y lectura, ya que se envía y se recibe los datos hacia el sensor por el mismo puerto.

Los puertos a utilizar:

```
#define DHT22_PIN 7
#define DHT22_VCC 7
```

El puerto D7, es donde se va a trasmitir y se va a recibir la cadena de bits para luego convertir de hexadecimal a decimal.

```
DHT_DDR |= (1<<DHT); //puerto salida
DHT_PORT |= (1<<DHT); //en alto(HIGHT)
```

Una vez definido el puerto para el DHT22, se prosigue con las funciones de envío y recepción.

Como la recepción de datos va a ser por un poleo de datos, no hace falta configurar ninguna interrupción.

En caso de que el modulo nos de error, se reiniciara a través del PC7, siendo este el puerto por el cual se alimenta el módulo.

10. Pluviómetro

El instrumento de medición pluvial es un sensor del tipo pivote, que es el más utilizado hoy en día en cualquier estación meteorológica.

Es un elemento que tiene gran facilidad para conseguirse y está muy probado.

Tiene un switch que actúa magnéticamente (reed switch), el cual, ante el pasaje de un imán a través del mismo este se pone en corto y se vuelve a abrir cuando termina de pasar.

La forma de este tipo de sensor es una ampolla de vidrio sellada la cual internamente tiene dos contactos muy próximos de material ferromagnético de níquel.

Figura 24 - Reed Switch.

La forma de funcionamiento es muy simple, actúa ante la proximidad de un campo magnético provocando que ambos elementos entren en contacto, dándole cierre al circuito.

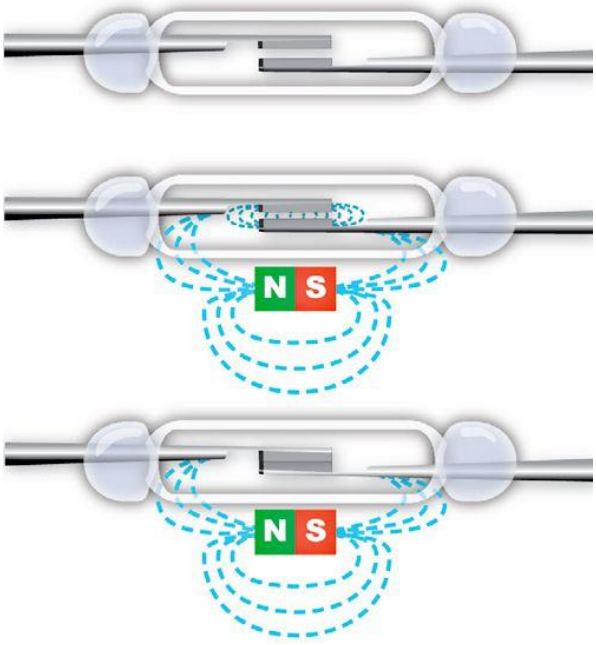


Figura 25 - Funcionamiento de Reed Switch.

El valor mínimo del campo magnético, para lograr hacer funcionar este sensor, es de 50 Gaus, con un pequeño imán permanente nos da la suficiente potencia para activarlo, es un sistema bastante seguro y puede utilizar cualquier tipo de tensión.

10.1. Cálculo volumétrico del sistema medidor

Para comprender su funcionamiento se realiza un análisis del módulo.

Figura 26 - Pluviómetro

La hoja de datos, informa que este medidor da un valor por cambio de posición, o sea, por cada volcado del medidor interno de 0.3mm de altura lluvia.

Entonces, a los 10 cambios de posición del balancín interno (tics) daría 3mm de lluvia, 55 tics = 15mm de lluvia y 100 tics = 30 mm de lluvia.

Si la superficie de medición, es de 5cm por 11cm, el cual, si se multiplica 5cm por 11cm daría una superficie de 55cm^2.

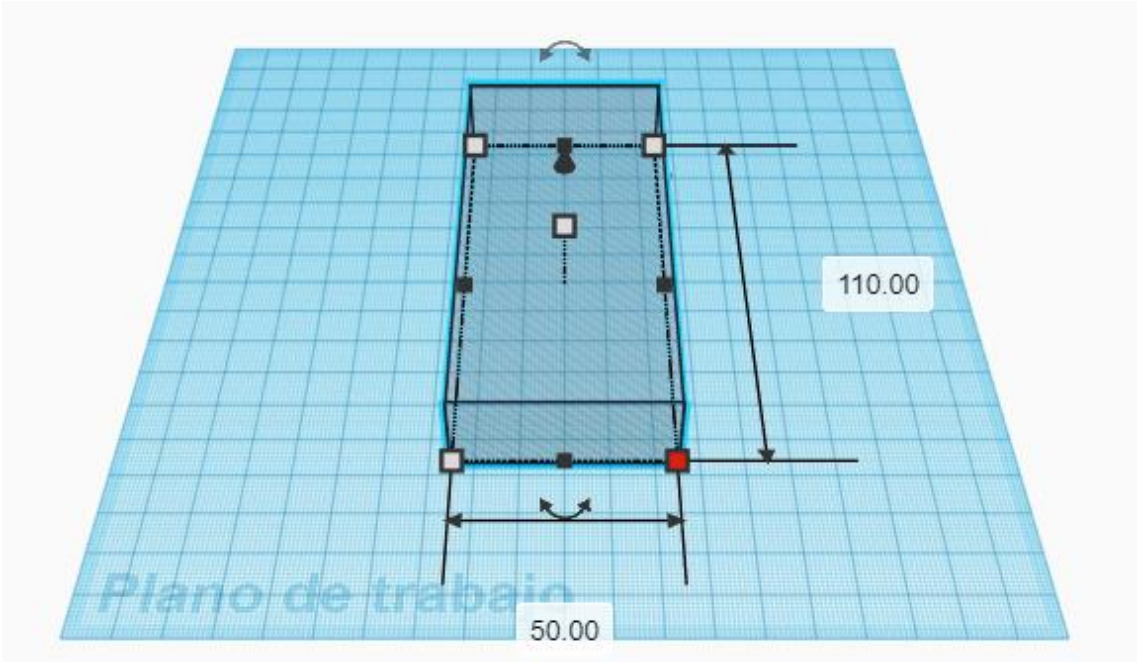


Figura 27 - Simulación de recipiente del pluviómetro.

Como primera premisa se puede decir que:

Altura de precipitación = lluvia recolectada / área de captación

Sabiendo que 1ml es 1cc o sea, 1ml=1cm^3.

Supongamos que se quiere medir 10ml de agua con un recipiente graduado, entonces:

$$\frac{10ml}{55cm^2} = \frac{10cm^3}{55cm^2} = \frac{2}{11} \cong 0.1818cm$$

La altura que debería tener el agua en el recipiente es de 0.1818cm, sabiendo esta altura y sabiendo la graduación que da el sensor, entonces h=0.1818cm cuantos tics debería dar:

$$\frac{0{,}3mm}{1{,}818mm} \rightarrow \frac{1tic}{x\_tics} = 6tics$$

Para la prueba experimental se realizó con un volumen mayor para poder tener una mejor precisión.

Se va tirando en el depósito del pluviómetro lentamente $50cm^3$ de agua.

Si la superficie es de $55cm^2$ la altura que nos debería dar es:

$$\frac{50cm^3}{55cm^2} = \frac{10}{11} \cong 0{,}9090cm = 9{,}090mm$$

Como hay veces que uno de los pequeños recipientes internos del pivot quedo lleno o no, puede ser que la medición no sea exacta, y de un vuelco de más.

Como los bordes del recipiente no son rectos, sino curvos, verificamos la medición por el método experimental.

Vaso medidor (cc)	Tics	Promedio
50	28	28,2307692
50	28	
50	28	
50	29	
50	28	
50	28	
50	29	
50	28	
50	28	
50	28	

50	28	
50	29	
50	28	

Tabla 2 - Valores de medición.

Entonces:

Como dio 28,23 vuelcos en promedio y la altura nos dio 9,090mm por cuentas. Se divide la altura en mm por la cantidad de vuelcos (((10/11)*10)/28,23)≅ 0,32; Este es el valor, que está dando en mm por cada vuelco del medidor.

$$\frac{\frac{50}{55}}{28{,}23} = 0{,}322 => tomo\ como\ valor\ 0{,}32$$

Se define en el conversor de lluvia en el servidor la medición de 0,32 mm por vuelco.

10.2. Cambio de estado (switch) y eliminar rebotes (debounce)

La conexión por detección de flaco descendiente que se explicó anteriormente es para detectar el cambio de estado y poder captar cuando el pluviómetro da un vuelco.

Dos inconvenientes:

1) Ante el cambio de estado, se producen mini rebotes que tienen que ser eliminados para una correcta medición, para esto se pone un capacitor que filtre estos falsos contactos.
2) Cuando se cierre el switch al estar con un capacitor, se descarga el mismo por este switch haciendo que circule mucha corriente por el mismo, esto produce un mayor desgaste de los contactos haciendo que se deteriore rápidamente, para evitar esto se coloca una resistencia para regular la corriente.

10.3. Rebotes (debounce)

El problema más común asociado a los contactos o relés mecánicos es el rebote, este se produce cuando dos contactos metálicos son obligados a chocar entre sí.

Mientras chocan entre sí se produce una transición de separación entre ambos contactos, se produce el efecto de una pelota rebotando cuando cae al piso, la cual rebota hasta detenerse.

Hay que asegurar el estado de encendido o apagado, el estado lógico.

En las siguientes figuras se muestra la transición de un contacto producido en su cambio, el estado lógico en el sistema.

Este primer estado de cambio suele utilizarse para un tipo de estado denominado pull_down, cuando está activo pasa al estado lógico 1 y cuando se inactiva al estado lógico 0 nuevamente.

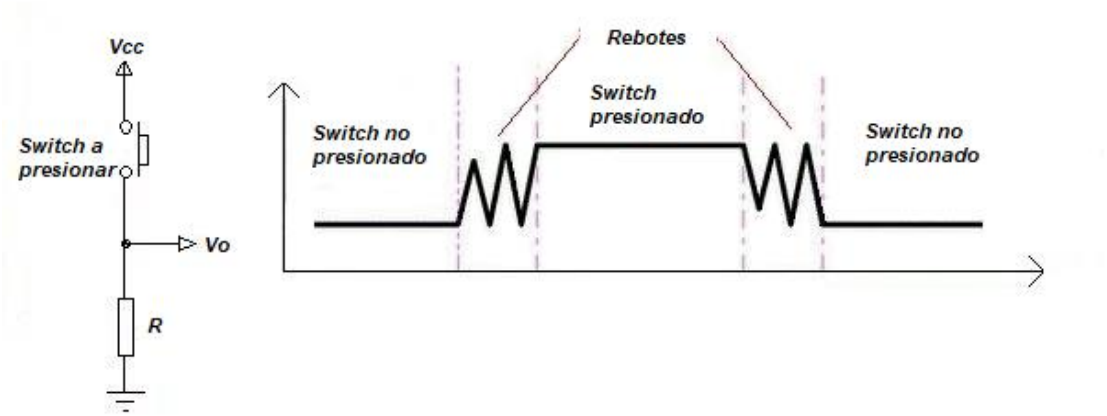


Figura 28 - Switch en modo pull-down.

Este segundo estado de cambio suele utilizarse para otro tipo de estado denominado de pull_up, cuando está activo pasa al estado lógico 0 y cuando se inactiva al estado lógico 1 nuevamente.

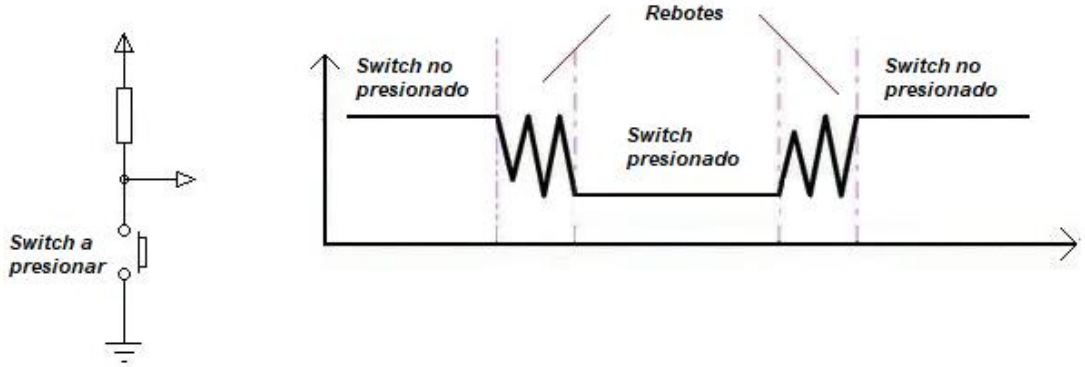


Figura 29 - Switch en modo pull-up.

Este último modelo es el utilizado, es el de pull_up.

Para lograr filtrar los cambios rápidos que se producen en la transición, es utilizado un capacitor con una resistencia, haciendo un filtro RC.

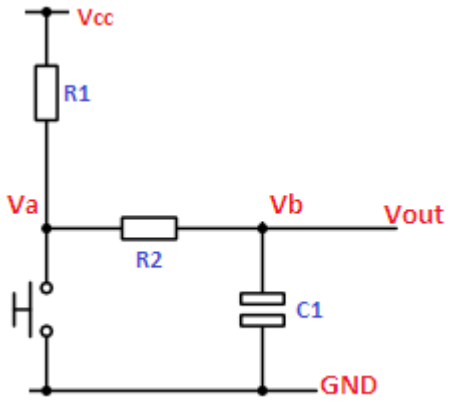


Figura 30 - Filtro RC para eliminar rebotes.

El circuito RC en este ejemplo es con dos resistencias y un capacitor.

- En la primera instancia el circuito está abierto, la tensión en el capacitor que inicialmente es cero, ahora se cargara por Vcc a través de R1 + R2.

- En la segunda instancia el circuito está cerrado, el condensador se descarga a cero por la resistencia R2 a masa, la cual limita la corriente que pasa por el contacto.

Para mejorar la carga del capacitor y volver al estado inicial rápidamente, se mejora el sistema poniendo un diodo en paralelo con R2, haciendo que el sistema se cargue por R1 únicamente.

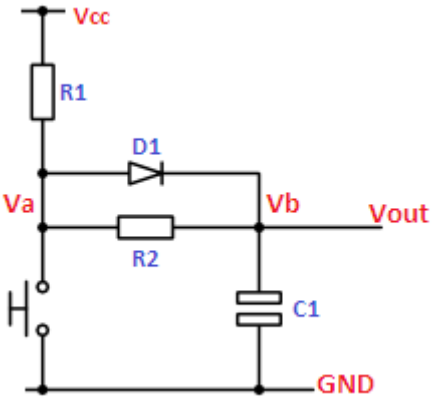


Figura 31 - Filtro RC mejorado para eliminar rebotes.

Haciendo un análisis de estado de contacto abierto ha cerrado se pueden ver en las siguientes figuras:

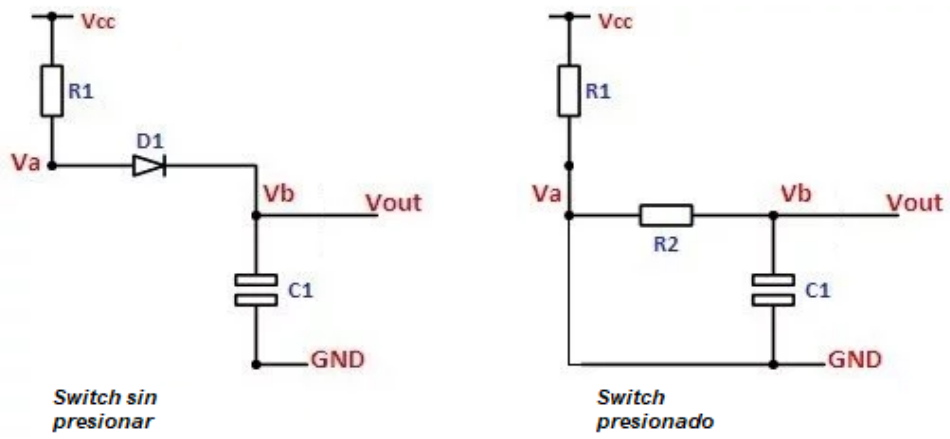


Figura 32 - Filtro RC mejorado en modo abierto y cerrado.

10.4. Cálculo del capacitor y resistencias

Siendo el voltaje inicial 5v y calculando un tau tal que τ=RC, esta constante se define en segundos.

La importancia de este cálculo, es cuánto tarda en cargarse el capacitor, se entiende que si tenemos un tau, es lo que tarda en cargarse el capacitor un 63,2% y con dos veces tau se carga un 86%.

Para la descarga del capacitor con un tau se descarga el 37% y con dos tau 13,5%.

Esto demuestra que un capacitor nunca estará cargado o descargado en su totalidad. Lo importante en este valor, es que para que un tau alcance el valor lógico que estamos necesitando.

Las ecuaciones de carga y descarga del capacitor son las siguientes.

$V(s) = V_s\left(1 - e^{-\left(\frac{t}{RC}\right)}\right)$	$V(s) = V_s e^{-\left(\frac{t}{RC}\right)}$
Carga del capacitor	Descarga del capacitor

Tabla 3 - Formulas de carga y descarga de filtro RC.

Donde V_s es la tensión de carga del capacitor, *t* es el tiempo transcurrido y *e* es la constante de Euler.

Con la ayuda del Matlab se visualiza el cálculo y se grafica para ver el valor correcto.

La versión de Matlab R2019a Trial con el pack de Symbolic Math Toolbox.

10.4.1. Descarga del capacitor

Se propone una resistencia R2=10kohms con un capacitor de 10microFaradios.

En la figura, R1 queda en corto, la cual no se incluye en el cálculo.

```
%Circuito descarga RC
clear all;
close all;
clc
%Parámetros del circuito
R=10e3; %Resistencia (10kOhms)
C=10e-6; %Capacitor (10MicroFaradios)
tau=R*C; %Constante de tiempo
Vs=5; %Capacitor inicial voltaje t=5.
Tiempo=0:tau/100:5*tau; %Sampleo de tiempo
Vc=Vs.*exp(-Tiempo./tau).*heaviside(Tiempo);

%Graficando el resultado
plot(Tiempo,Vc)
xlabel('Tiempo (s)')
ylabel('Amplitud(V)')
title('V_C(Capacitor)')
%Fin
```

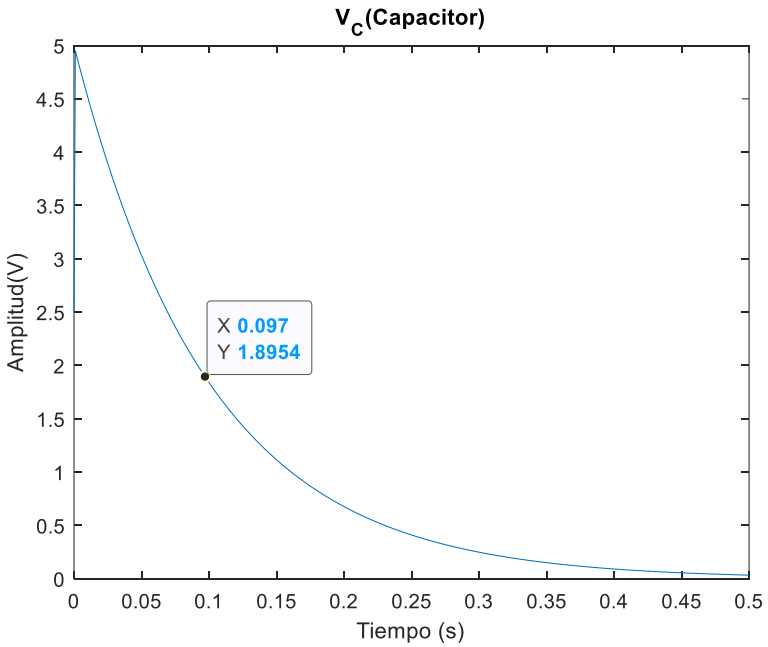


Figura 33 - Descarga del circuito RC.

En 10ms tenemos el 37% del valor que necesitamos para que nos dé un valor correcto.

Como los rebotes se producen en el orden de los microsegundos, va a estar sobrado de tiempo en la transición asegurando el valor correcto.

Cualquier rebote que pueda ocurrir, es filtrado por el capacitor.

La corriente que circula por el interruptor magnético es de:

$$i = \frac{v}{R} = \frac{5v}{10^3 ohms} = 5^{-4} A$$

Es una corriente muy pequeña, asegurando la durabilidad del sistema.

10.4.2. Carga del capacitor

Se propone una resistencia R1=10kohms, R2=10kohms, C=10microFaradios.

```
%Circuito RC
clear all;
close all;
clc
%Parámetros del circuito
R=20e3; %Resistencia (10kOhm)
C=10e-6; %Capacitor (10microFaradios)
tau=R*C; %Constante de tiempo
Vs=5;   %Tensión de entrada
Time=0:tau/10:5*tau; %Sampleo del tiempo
V_C=(Vs).*(1-exp(-Time./tau)).*heaviside(Time);
%Graficando el resultado
plot(Time,V_C)
xlabel('Tiempo (s)')
ylabel('Amplitud (V)')
title('V_C(Capacitor)')
%Fin
```

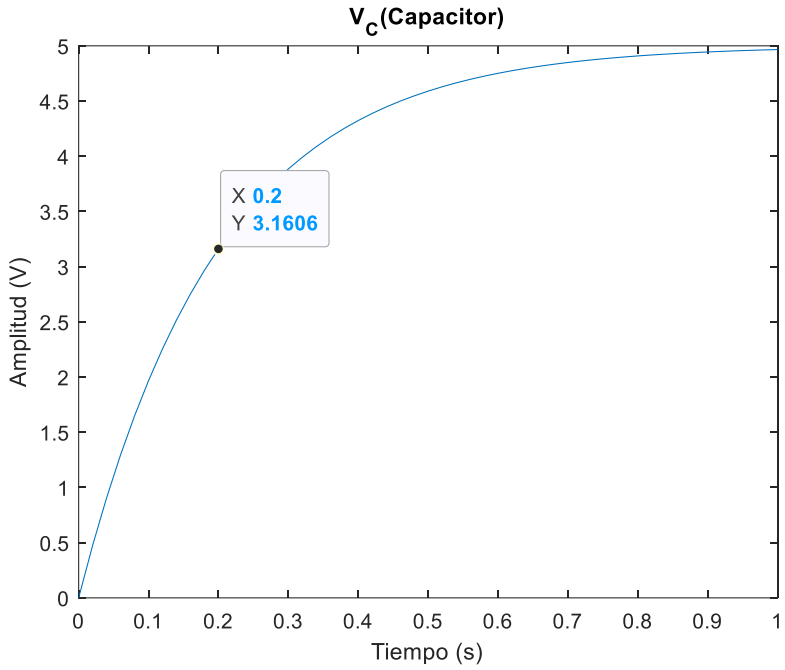


Figura 34 - Carga del circuito RC.

Se observa en la figura, que para alcanzar el valor de recuperación del 63% para llegar a los 5v es más lento que en el caso anterior debido a las dos resistencias, para mejorar ese tiempo, puenteamos la resistencia R2 con un diodo, haciendo que circule libremente la corriente cuando se vuelve a cargar, el código de matlab queda de la siguiente forma.

```
%Circuito RC
clear all;
close all;
clc
%Parámetros del circuito
R=10e3; %Resistencia (10kOhm)
C=10e-6; %Capacitor (10microFaradios)
tau=R*C; %Constante de tiempo
Vs=5;   %Tensión de entrada
Td=7e-1; %Caída de tensión en el diodo
Time=0:tau/10:5*tau; %Sampleo del tiempo
V_C=(Vs-Td).*(1-exp(-Time./tau)).*heaviside(Time);
%%Graficando el resultado
plot(Time,V_C)
xlabel('Tiempo (s)')
ylabel('Amplitud (V)')
title('V_C(Capacitor)')
%Fin
```

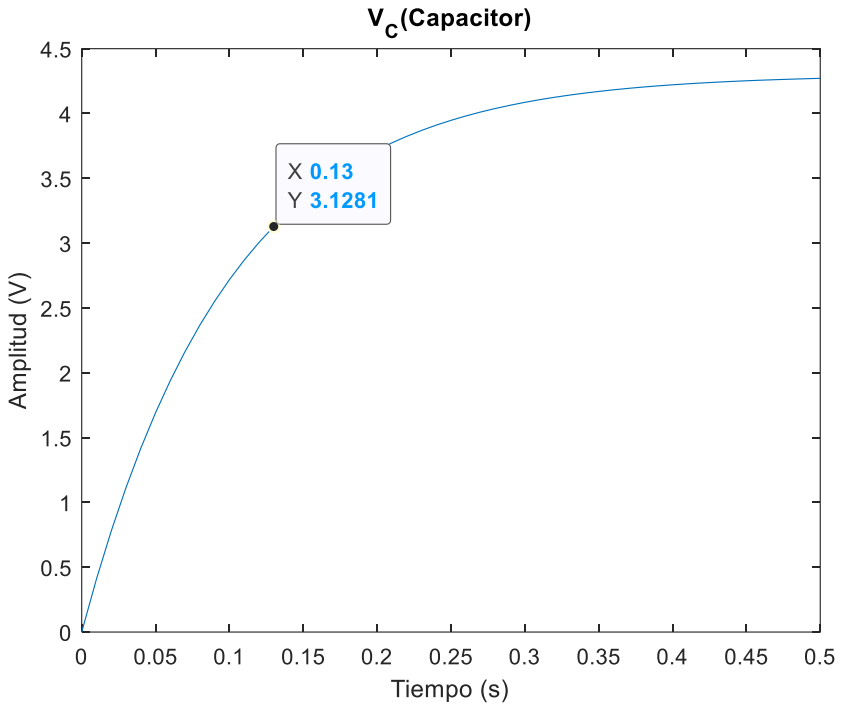


Figura 35 - Carga del filtro RC mejorado.

Mejora la recuperación de la carga del capacitor, cuando llegue a la diferencia de 0,7v del diodo se sigue cargando a través de la resistencia R2.

El circuito se implementó en una placa experimental.

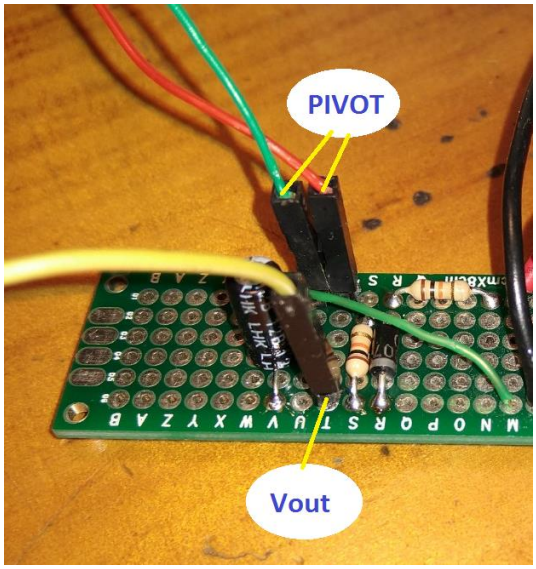


Figura 36 - Circuito de Pivot.

10.5. GPRS

Toda la información es enviada al servidor para su análisis por GPRS.

El GPRS SIM800L que es utilizando es para disminuir espacios y abaratar costos, es una placa del fabricante RoHS.

El intervalo de tiempo que se utilizara para envío de datos será cada 10minutos.

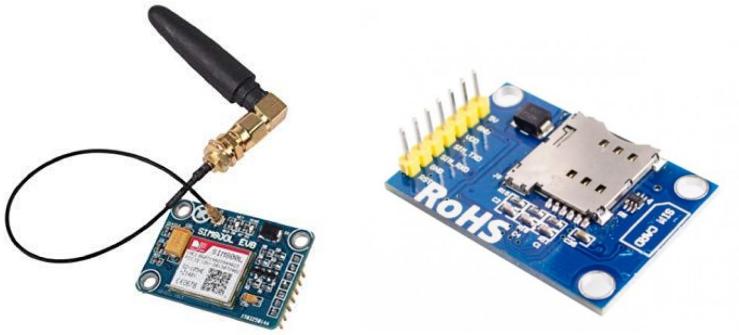


Figura 37 - Modulo GPRS - RoHs.

Cuando está trasmitiendo suele alcanzar los 500m A pico según la hoja de datos, esta información es utilizada para dimensionar la fuente.

Tiene una antena instalable para gabinete, lo que es muy útil.

El chip del GPRS es un chip SimCom 800L funciona con niveles de tensión de 3.3v, la placa ya resuelve este inconveniente funcionando todo en 5v.

La placa diseñada por RoHS no tiene todas las opciones que puede proporcionar el chip SimCom800L.

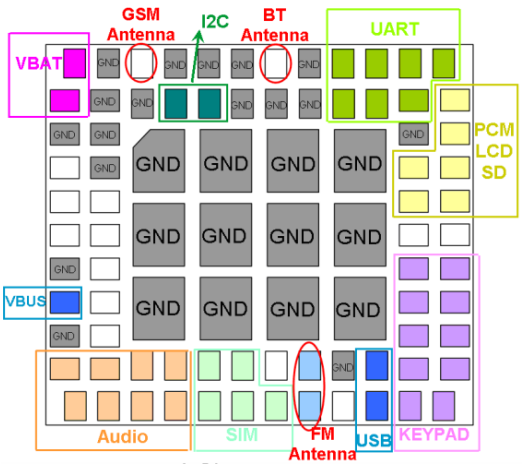


Figura 38 - Pinout SimCom800.

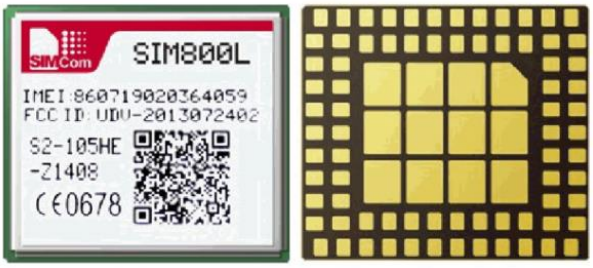


Figura 39 - Chip Sim800L.

Trae deshabilitado el KEYPAD, Audio, VBAT, I2C, Audio, PCM, LCD, SD.

Este tipo de sistemas funciona con un tipo de código vía comandos AT a través del puerto UART.

La velocidad de comunicación establecida es de 2400baud, cuando empieza a comunicarse, el modulo del GPRS verifica la velocidad según los datos recibidos.

Al inicio se envían datos AT y comienza la comunicación con el microcontrolador, el primer inconveniente a resolver, es que por default, estos chips, vienen activados con el modo ECO, el cual, cuando es recibido un dato, devuelve el dato que mandamos más la respuesta del dispositivo.

Como el modo ECO suele complicar el sistema, es anulado el ECO apenas comienza a inicializar todo el sistema.

El comando utilizado es “AT+ ATE0” y el carácter '\r'.

Es activada la entrada de reinicio (RST) debido a que, muchas veces no reacciona la placa cuando está funcionando durante mucho tiempo.

La geolocalización se toma una vez al día, para una disminución de datos, como así también que el proceso total del sistema sea más rápido, debido a que la geolocalización es muy lenta.

10.6. DHT22

Hay dos tipos de sensores de temperatura y humedad al alcance del presupuesto, este es el DHT11 y el DHT22, este último el doble de precio que el primero.

Ambos sensores son del tipo capacitivo.

En el siguiente cuadro comparativo podemos ver las ventajas de uno frente al otro.

DHT11	**DHT22**
Temperatura	
0°C - 50°C +/- 0.2 °C	-40°C - 80°C +/- 0.5 °C

Humedad	
20% - 90% +/- 4%	0% - 100% +/- 2%
Ciclo de lectura	
1seg	2seg

Tabla 4 - Comparativa DHT11 - DHT22

Se puede ver reflejado en la tabla las prestaciones de uno frente al otro.

En las pruebas realizadas, el DHT11 se tenía que reiniciar debido a la falta de respuesta del módulo, lo cual reafirmo en este trabajo la decisión de dejar en funcionamiento el DHT22.

El puerto PC7 es de alimentación, en el caso de dar error, se pueda reiniciar el modulo.

Los pines de conexionado son los siguientes, además se utilizó la resistencia interna de pull-up del microcontrolador.

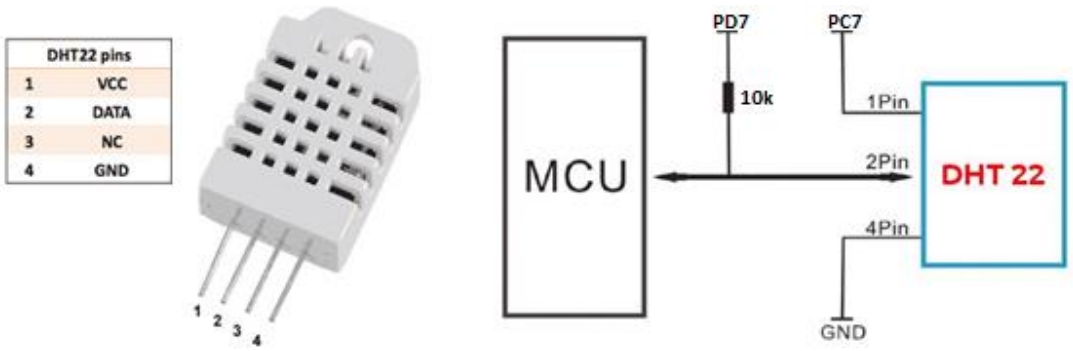


Figura 40 - Diagrama de conexiones del DHT22

Los datos recibidos del DHT22, son en hexadecimal, se convierte en un valor decimal.

La resistencia de pull-up que muestra la figura, es la misma que tiene internamente el micro controlador.

El sensor es apagado cada vez que el dispositivo tira un error, ya sea, porque el checksum da un valor incorrecto o por que no responde.

El consumo es muy bajo en estado en reposo.

Los puntos a tener en cuenta para el sensor:

1) Sensibilidad a la radiación ultravioleta, tiene que estar protegido.
2) Calidad del cable, tanto para enviar tensión como el que envía los pulsos de comunicación, es muy sensible, se utilizó cable mallado.
3) Control de errores, por lo que no debería devolver datos con error, pero, por si las dudas, le dejamos el control de errores, caso que si falla, el sistema lo reinicie.

Diagrama de flujo del DHT22 proporcionado por el fabricante.

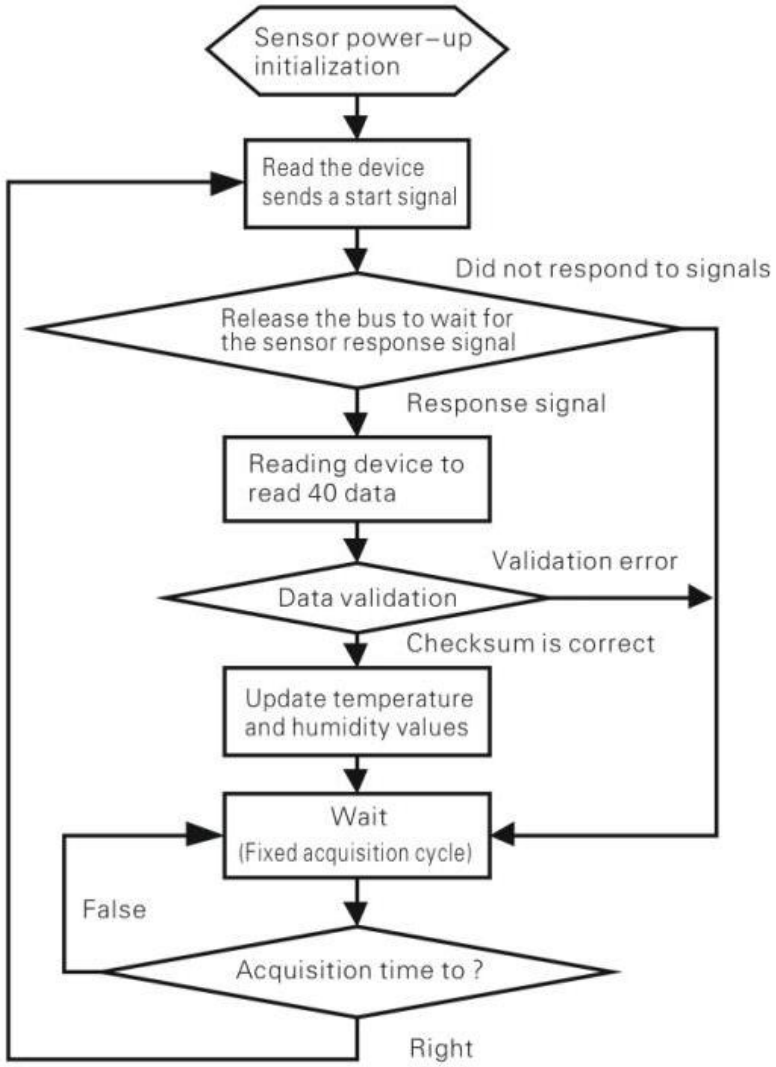


Figura 41 - Diagrama de flujo del DHT22.

Como en el DHT22 tiene un solo canal de comunicaciones, el manejo de tiempos se especifica en la siguiente figura según el fabricante:

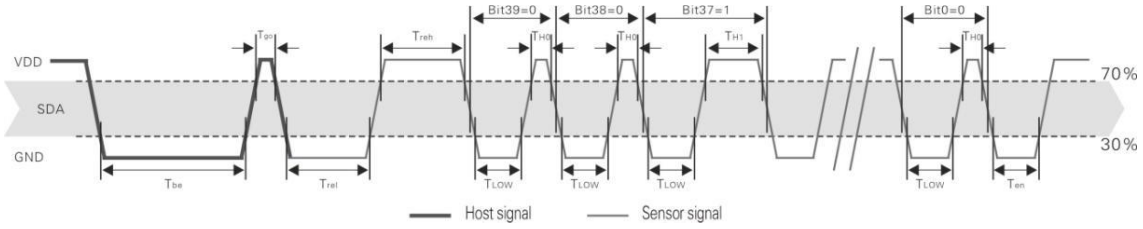


Figura 42 - Tiempos de conmutación y tiempos de los bits en el canal de comunicación.

Para comprender como es el canal de comunicación, el siguiente cuadro explica cada uno de los parámetros.

Símbolos	Parámetros	Mínimo	Típico	Máximo	Unidad
Tbe	Tiempo de inactividad antes del inicio(MCU)	0.8	1	20	*mS*
Tgo	Pongo en estado alto(high) y espero por una respuesta.(MCU)	20	30	200	*µS*
Trel	Tiempo de espera de la respuesta en estado bajo(low).(DHT22)	75	80	85	*µS*
Treh	En estado alto(high) para responder.(DHT22)	75	80	85	*µS*
TLOW	Señal "0", "1" para empezar envío de datos.(DHT22)	48	50	55	*µS*
TH0	Señal en "0" tiempo largo, para separar bits.(DHT22)	22	26	30	*µS*
TH1	Señal en "1" tiempo largo, para separar bits.(DHT22)	68	70	75	*µS*
Ten	Avisa que ya se enviaron todos los datos.(DHT22)	45	50	55	*µS*

Tabla 5 - Manejo de tiempos en el canal de comunicaciones del DHT22.

La programación realizada fue:

1) Reseteo del puerto, y espera de 100ms
2) Poner estado bajo(LOW) el puerto PD7 por 500us
3) Poner estado alto(HIGH) el puerto PD7 por 40us
4) Modo espera para respuesta del DHT22 por 80us
5) Modo recepción del DHT22 por 80us (empieza a trasmitir datos el DHT22)
6) Recibe 16 bits, demora por 30us
7) Recibe 16 bits
8) Recibe 8 bits y guardado todo en un solo string.

Este módulo es utilizado por polling (poleo).

La señal que es recibida está compuesta en dos partes, la primera de 16bits para la humedad y la segunda de 16 bits para la temperatura. Se reciben 8 bits para la verificación, dando un total de 40 bits.

Teniendo todo dos los datos obtenidos, se pasa a verificar los datos, si son correctos, y la suma de los binarios es igual al checksum,

Procedemos a pasar los datos hexadecimales en decimales y guardarlos en la variable que vamos a enviar a nuestro servidor.

Si se verifico la suma, si es correcto, devuelvo el valor, en caso contrario reinicio el DHT22.

10.7. Sistema de energía

Hay diferentes tipos de paneles solares, desde diseños a usos, tanto de bajo consumo como de alto consumo, algo a tener en cuenta a la hora de comprar, es que sea para exteriores y que venga blindado cono un vidrio protector.

Fue comprado un panel solar de 10W/h, con una batería de 12v 7A/h con un regulador de carga con salida de 12v de 10A.

Figura 43 - Sistema de alimentación de energía.

Se instalará un panel solar de 10Wh con un soporte para fijarlo.

Panel solar de 10W de Pmax.(289*275*17mm)

Figura 44 - Panel solar 10W (Max)

Batería de plomo marca MOURA de 12V 7Ah/20Hs.

Figura 45 - Batería MOURA 7Ah/20h.

Controlador solar programable de 12/24v 10A(máximo) de la marca Smart.

Figura 46 - Regulador de paneles solares para carga de baterías.

Debido a que las salidas de los controladores programables encontrados en el mercado, tienen una salida de 12v, se colocó un conversor de tensión de 12v a 5v, para este fin se utilizó un LM2596-5v.

El LM2596-5v es una fuente switching, por consiguiente tiene bajas perdidas de conversión.

El LM2596-5v soporta hasta 3A, logrando obtener una fuente que nos pueda proporcionar buena potencia.

Se armó lo establecido en el datasheet:

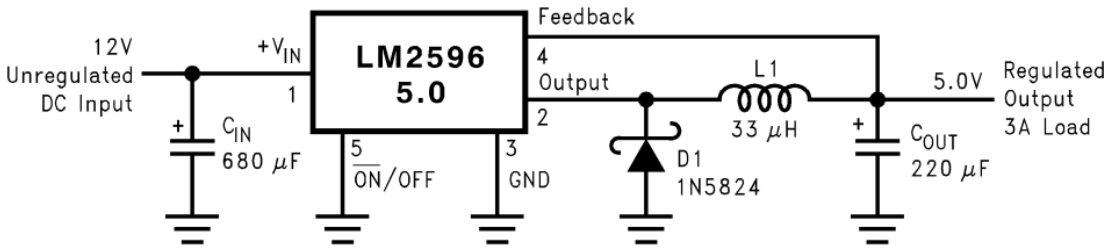


Figura 47 - Esquema LM2596-5.0v

Se armó la fuente en una placa experimental, se prefirió armar la fuente una placa alejada del sistema de control.

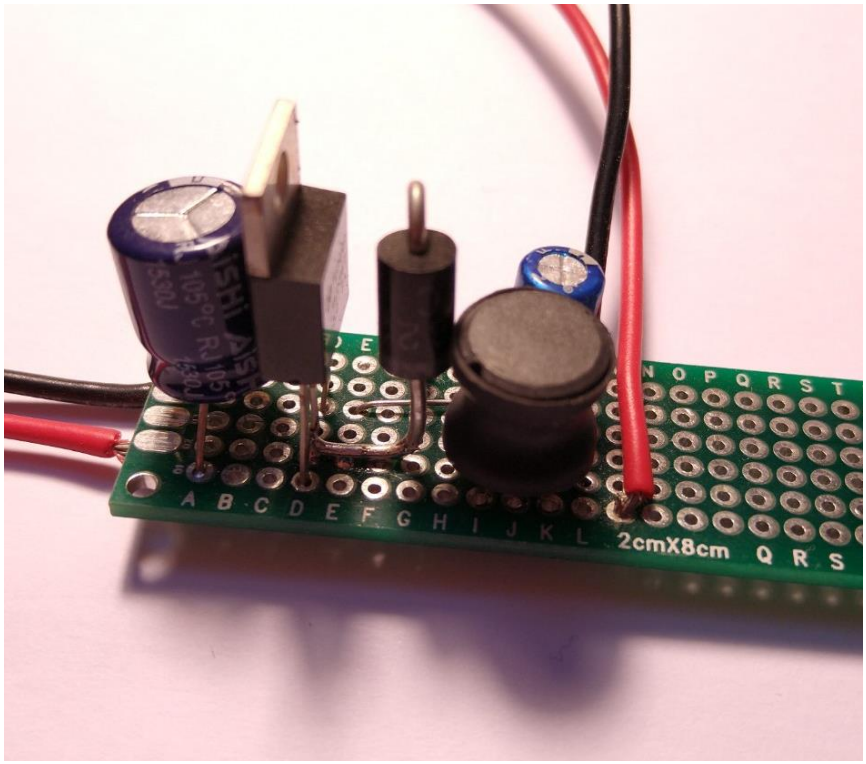


Figura 48 - Circuito de fuente LM2596-5.0v.

10.8. Cálculo de consumos en nuestro sistema

Los sistemas de mayor consumo hoy en día son los basados en redes celulares y comunicaciones satelitales.

Se realizó un análisis de consumo para dimensionar el sistema de abastecimiento de energía.

10.8.1. Consumo Atmega128

Para el cálculo de potencia podemos sacar el máximo teórico de nuestro sistema, con el oscilador interno de 1MHZ, consumiendo a esta frecuencia unos 4mA alimentado con 5V.

Figure 162. Active Supply Current vs. Frequency (1 - 20MHz)

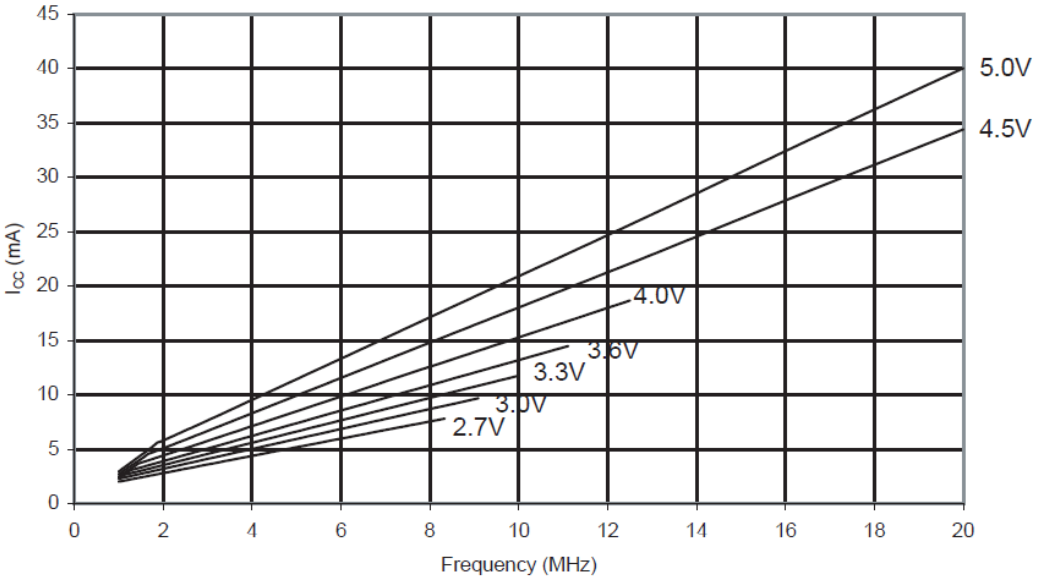


Figura 49 - Diagrama de Frecuencia-Consumo Atmega128.

El microcontrolador, tiene un consumo sin nada conectado menor a 5mA, que no aparenta ser mucho, pero si tenemos una batería de 1200mAh, y dejándolo prendido sin realizar ninguna función, podemos calcular el consumo, entonces:

1200mAh/5mA=240hs, dando 10dias

Lo que significa, que con una batería de 1200mAh solo funcionando, sin hacer nada nos duraría 10dias.

En la prueba de laboratorio la placa sola, sin ningún conexionado con un led en la placa con una resistencia de 1K (5mA) me da un consumo de 9,3mA – 5mA= 4,3 casi exacto con lo que decía el datasheet.

10.8.2. Consumo GPRS

Los equipos GPRS son los de más consumo, debido al sistema de transmisión de datos utilizado.

La tabla de consumos de este chip es:

- Consumo de corriente (máx): 500 mA
- Consumo de corriente (modo de reposo): 0.7 mA

Si tenemos una batería de 1200mAh/500mA=2,4hs, lo que significa, que andando continuamente, no llega ni a un día.

Por esta razón, cada vez que se tiene que transmitir, se saca del modo suspendido, funcionando por aproximadamente 30 segundos y volver al modo suspendido.

Los consumos son valores a tener en cuenta, para el cálculo de consumos se realizan las siguientes mediciones una por una y se suman.

10.8.2.1. Placa de RF

La de mayor consumo, se hace un análisis más exhaustivo.

SimCom800L (Placa RF):

Figura 50 - Consumo de modulo RF con Tester.

Después de varios intentos, el valor máximo obtenido es de 185mA, aunque, según el datasheet nos indica que el consumo máximo es de 500mA, lo cual, se realizó una prueba diferente.

Colocando una resistencia .1ohms (marrón negro plateado) y midiendo la caída de tensión en la misma con un osciloscopio:

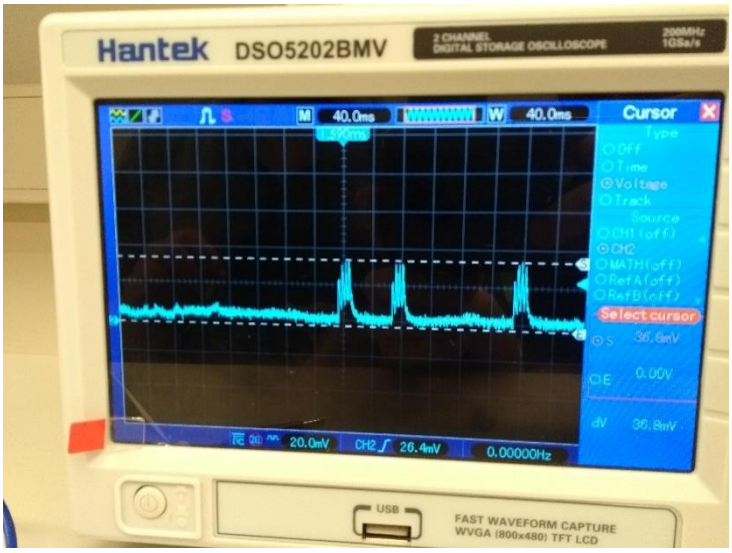


Figura 51 - Niveles de tensión den resistencia para medida de consumo.

Dando un consumo aproximado de 36.8/0.1(volt/ohm) = 368mA, el cual es muy próximo al valor dado en la hoja de datos, es probable que varié por la distancia entre antenas.

El tester nos da un valor diferente, debido a que este da un valor promedio máximo.

10.8.3. ConsumoDHT22

El consumo máximo es de 2,5mA, según la hoja de datos, en modo suspendido es menor a 1mA.

10.8.4. Suma de consumos

Micro controlador: 5mA (Suponiendo un casi 20% de más)
GPRS: 200mA durante 30seg, después se pone en modo suspendido 10mA
DHT22: 2,5mA durante 30seg, luego se pone en modo suspendido en 1mA.

Suma total en 24hs:

Atmega128: 24hs 5mA =>**120mAh en 24hs**

GPRS: 144veces de 200mA de 30Seg => 72 minutos en el día = 1,2 hs
144veces de 10mA durante el resto del día. => 22,8 hs
Dando un total de 200mA*1,2hs + 10mA*22,8hs = **468mAh en 24hs**

DHT22: 144 veces de 2,5mA de 30Seg = 72 minutos en el día => 1,2 hs
144 veces de 1mA de 22,8hs
Dando un total de 2,5mA*1,2hs + 1mA*22,8hs = **25,8mAh en 24hs**

Sumando todos los consumos en 24hs:

Consumo Total = 120mAh + 468mAh + 25,8mAh = 613,8mAh

Sumando todo nos da una batería con una capacidad de 613,8mAh.

El consumo es (613,8*5v)/1000 = 1093,8mWh = 1,0938Wh

Sobredimensionando el sistema en un 50%, el consumo en 24hs es de 1,6407Wh

La batería utilizada proporciona 7Ah y el panel tiene una capacidad máxima de 10Wh.

Teniendo 2Wh de potencia en 24hs, por un tiempo de una hora son 8,3mAh de consumo promedio.

Suponiendo un rendimiento del 10% debido a inclemencias del tiempo, lluvia y suciedad del momento, estaría proporcionando 1Wh, con lo que se deduce que con 2hs funcionando a esa capacidad se estaría alcanzando lo necesario para el consumo del día.

Para ver la implementación del todo el sistema se deja una variedad de fotos en el anexo 1.

11. Plataforma de Visualización y Registro de datos

11.1. Servidor

El servidor dará en forma práctica las mediciones tomadas, se utilizó una base de datos MySQL.

El sistema operativo utilizado es Linux, se utilizó debían 8 con kernel Linux 3.16.0-8-amd64 on x86_64.

11.2. Página web

El sistema web, está configurada con html y css con las plataformas graficas proporcionadas por Boostrap.

Boostrap es una plataforma con muchos ejemplos gratuitos, de donde se extrajeron ejemplos para configurar los gráficos de la página.

El direccionamiento del servidor fue al domino agroargentino.com.ar, el dueño de este dominio es Mariano Pablo Naboni, esto se puede verificar en Nic Argentina ingresando en nic.ar

11.2.1. Página principal

La portada principal solamente consta de una simple presentación:

Figura 52 - Pagina web principal.

11.2.2. Mapa

La opción de MAPA muestra las centrales que tenemos funcionando en la zona.

En “Reportes” dirige al reporte grafico donde muestra lo sucedido hasta el momento en el día de esa central.

Cada central tiene su propio nombre y cada uno da la temperatura y humedad máxima, así como también el acumulado de lluvia en el día.

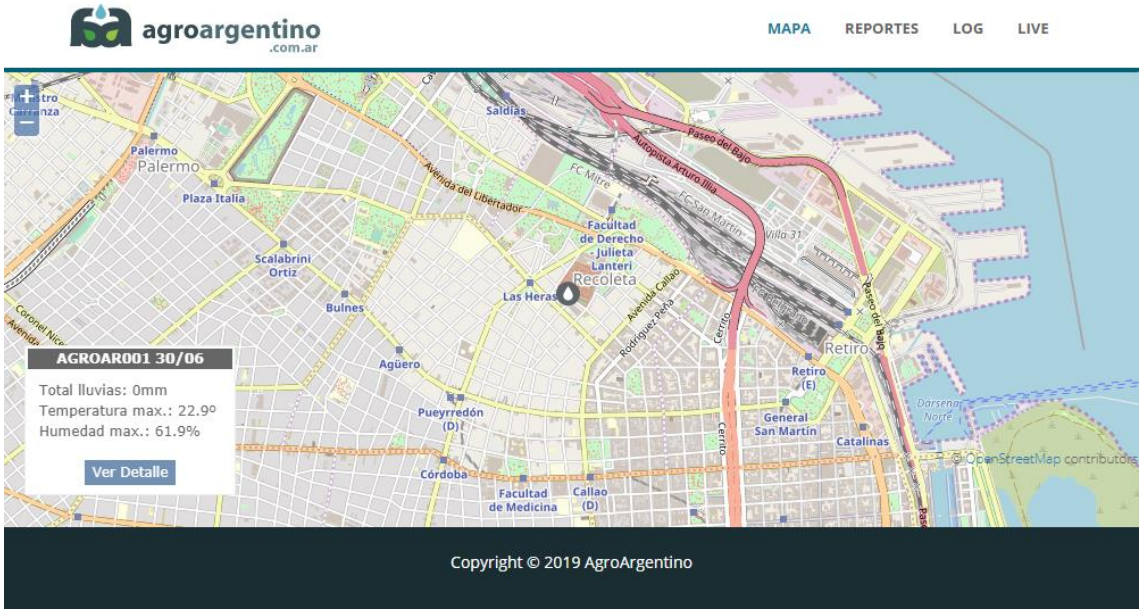


Figura 53 - Mapa web de Geolocalización.

Cuando se hace un clic sobre el mismo nos trae en una vista rápida de cuales fueron los últimos datos máximos obtenidos en el día.

Estos datos empiezan desde las 00:00hs, si se entra a esa hora, no va a mostrar datos hasta que empiece a recaudar información.

11.2.3. Reportes

En esta opción, se puede buscar por el nombre del equipo.

Con una rápida mirada, se ve qué cambios importantes hubo en el día, como así los máximos ocurridos.

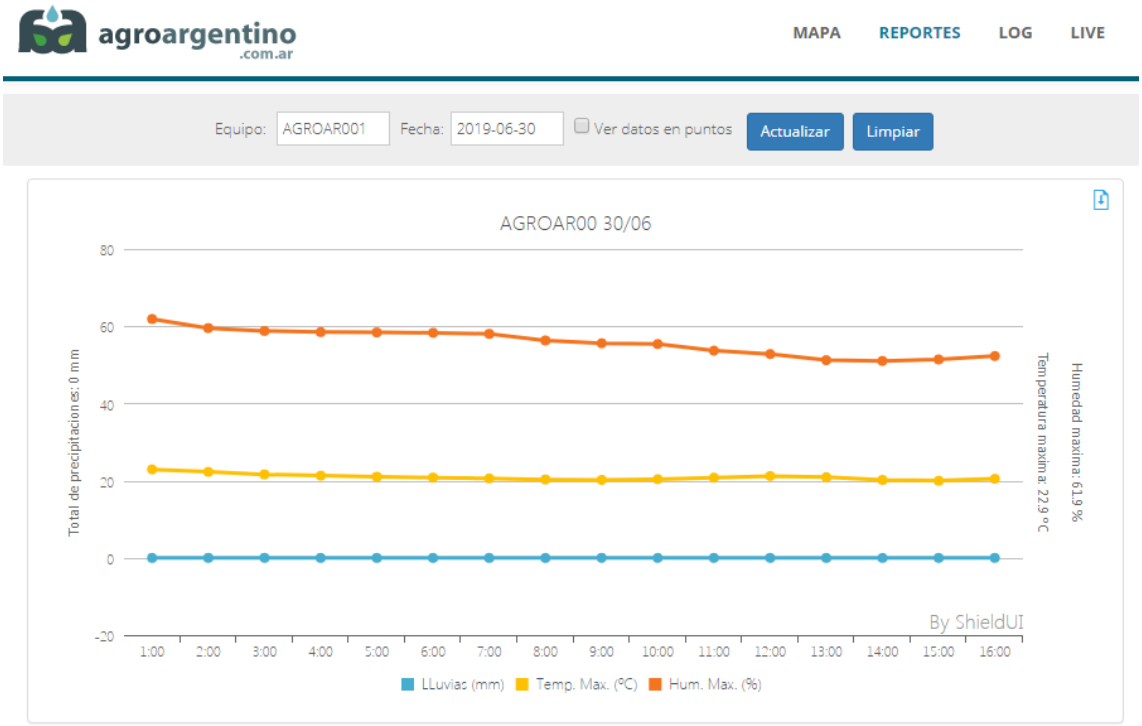


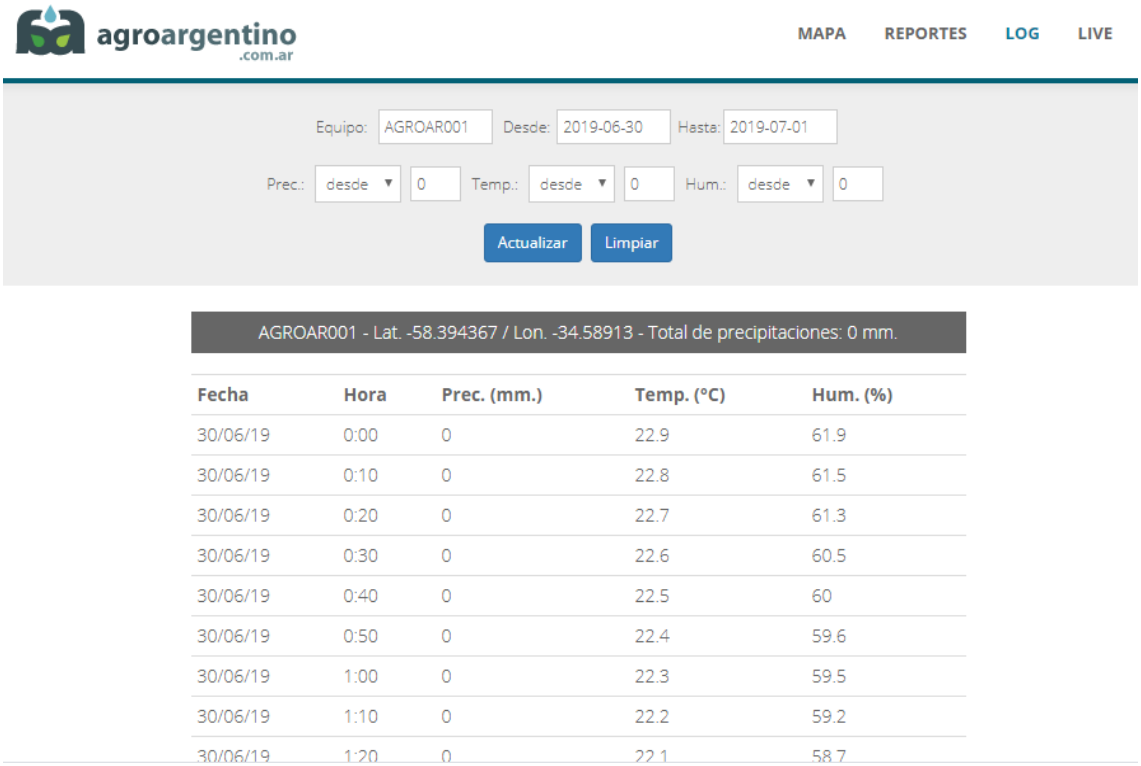
Figura 54 - Grafico histórico de lluvia, humedad y temperatura.

Se puede dar alguna alerta en el caso que fuese necesario.

En todos los gráficos hay un histórico hacia atrás, siendo esto muy útil a la hora de sacar conclusiones y toma de medidas a futuro.

11.2.4. LOG

A la hora de detallar un caso en particular, que puede haber pasado, se puede bajar la información y filtrar lo necesario.



Fecha	Hora	Prec. (mm.)	Temp. (°C)	Hum. (%)
30/06/19	0:00	0	22.9	61.9
30/06/19	0:10	0	22.8	61.5
30/06/19	0:20	0	22.7	61.3
30/06/19	0:30	0	22.6	60.5
30/06/19	0:40	0	22.5	60
30/06/19	0:50	0	22.4	59.6
30/06/19	1:00	0	22.3	59.5
30/06/19	1:10	0	22.2	59.2
30/06/19	1:20	0	22.1	58.7

Figura 55 - Histórico de lluvia, humedad y temperatura.

La información se puede analizar detenidamente con una planilla de cálculos y replantear nuevos desafíos.

Cada equipo, lleva su nombre particular, lleva una ubicación geo localizada por el mismo equipo, si se mueve por algún caso en particular, a través del nombre se sabe dónde está ubicado.

11.3. Registros o database

Como se utiliza una base de datos, se implementa la inserción de los datos a través de un GET.

Esta opción fue implementada, debido a que no realizan muchos envíos de datos y resultar más práctico a la hora de utilizarlo.

Dado que se configuro el envío a través de un GET, cada vez que se envía un dato, se tiene que ver, si los datos que se van a guardar, son datos verdaderos y no un intento de tirar abajo la base de datos, para eso se hace un análisis de los datos recibidos.

Una vez analizado los datos se hace un INSERT para guardar los datos en la base de datos.

También se hace un análisis de la posición, ya que como la geolocalización dada por el GPRS no es muy exacta, verificamos que este dentro del rango de +/- 0.002 del valor anterior, si esta en este rango, no cambio el nuevo valor de geolocalización del equipo, esto es por si se llega a mover por algo en especial, o lo cambiaron adrede.

El envío de datos es cada 10 minutos, se realiza una verificación de datos nuevos cada 5 minutos en el servidor, si hay un dato nuevo, se guarda el nuevo valor.

Se realiza muchas veces, ya que el equipo puede llegar a tardar en dar red, ya que todo el tiempo tiene que estar verificando si está establecida la conexión en caso contrario se vuelve a enviar nuevamente la re conexión.

12. Conclusiones

12.1. Conclusiones generales

Es un sistema autónomo, desarrollado a partir de la necesidad que tiene el productor agrario de estar actualizado en lo que está ocurriendo en su zona de trabajo, permitiendo así la toma acertada de decisiones durante la campaña productiva.

El equipo está diseñado para tomar el registro de las lluvias, la humedad y la temperatura, que son datos esenciales a la hora de organizar la práctica de algún cultivo.

Tiene un historial en el cual queda el registro del día y hora de lo ocurrido, que se puede visualizar tanto en una tabla como en un sistema gráfico.

Se geo localiza y cambia su posición si este es corrido de lugar.

Al visualizar los datos en el momento que están ocurriendo permite sacar rápidas conclusiones optimizando así la toma de decisiones. Por ejemplo: si llovieron 20mm, durante la noche, es propicio para poder hacer una siembra, a primeras horas de la mañana en el caso de la soja. Muchas veces ante la falta de este dato, se pierden días para la nueva siembra.

A la hora de tomar una determinación para fumigar, es importante saber si llovió y cuanto en la zona a tratar, debido a que en la hoja que tiene agua se malogra el agroquímico y no es efectivo.

Todo eso es un ahorro de tiempo y de combustible, ya que muchas veces hay que ir al lugar a verificar como está el cultivo.

Es una herramienta eficaz, moderna y sobre todas las cosas, está en la nube, algo sumamente utilizado hoy en día y verificable desde cualquier dispositivo móvil.

El sistema desarrollado cumplió las expectativas planteadas.

12.2. Conclusión particular

Es un sistema que es de fácil armado, con módulos que se pueden conseguir fácilmente en Argentina.

Si se necesita reemplazar alguna parte por rotura o falla es de bajo costo.

Es de fácil instalación, se pone en el lugar, se le configura un nombre y se agrega directamente al sistema.

12.2.1. Puntos a mejorar

1) El manejo de velocidades del sistema como así los tiempos de suspensión de los módulos y microcontrolador.
2) Un sistema de gráficos en la página para ver en un pantallazo cómo fue cambiando el tiempo a lo largo del día, mes y año.
3) Poder guardar el histórico en una planilla de Excel para su posterior utilización en caso de ser necesario.
4) Un sistema de medición de viento y dirección, esto ayudaría también para cuando se quiere ir a fumigar, sobre todo de qué lado está el viento para saber si puede o no salir alguien afectado.

FUENTES DE INFORMACIÓN.

agter. (21 de 07 de 2019). *agter*. Obtenido de agter: http://www.agter.org/bdf/es/corpus_chemin/fiche-chemin-230.html

Apache. (9 de Marzo de 2019). Obtenido de https://www.apache.org/

BBC. (21 de 07 de 2019). *BBC*. Obtenido de New BBC: https://www.bbc.com/mundo/noticias/2014/04/140409_ciencia_agricultura_urbana_fao_diez_ciudades_america_latina_np

Bootstrap. (9 de Marzo de 2019). Obtenido de https://getbootstrap.com/docs/4.3/examples/

Cintero, S. (25 de 11 de 2018). *Sistema propio de seguimiento en vivo (GPS Tracker)*. Obtenido de http://www.saulcintero.com/seguimiento-en-vivo-gps-tracker/

CSS, H. y. (31 de Enero de 2019). Obtenido de Programo Ergo Sum: https://www.programoergosum.com/cursos-online/paginas-web/24-crea-tu-pagina-web-con-html5-y-css3/introduccion-web

Debian. (9 de Marzo de 2019). Obtenido de https://www.debian.org/index.es.html

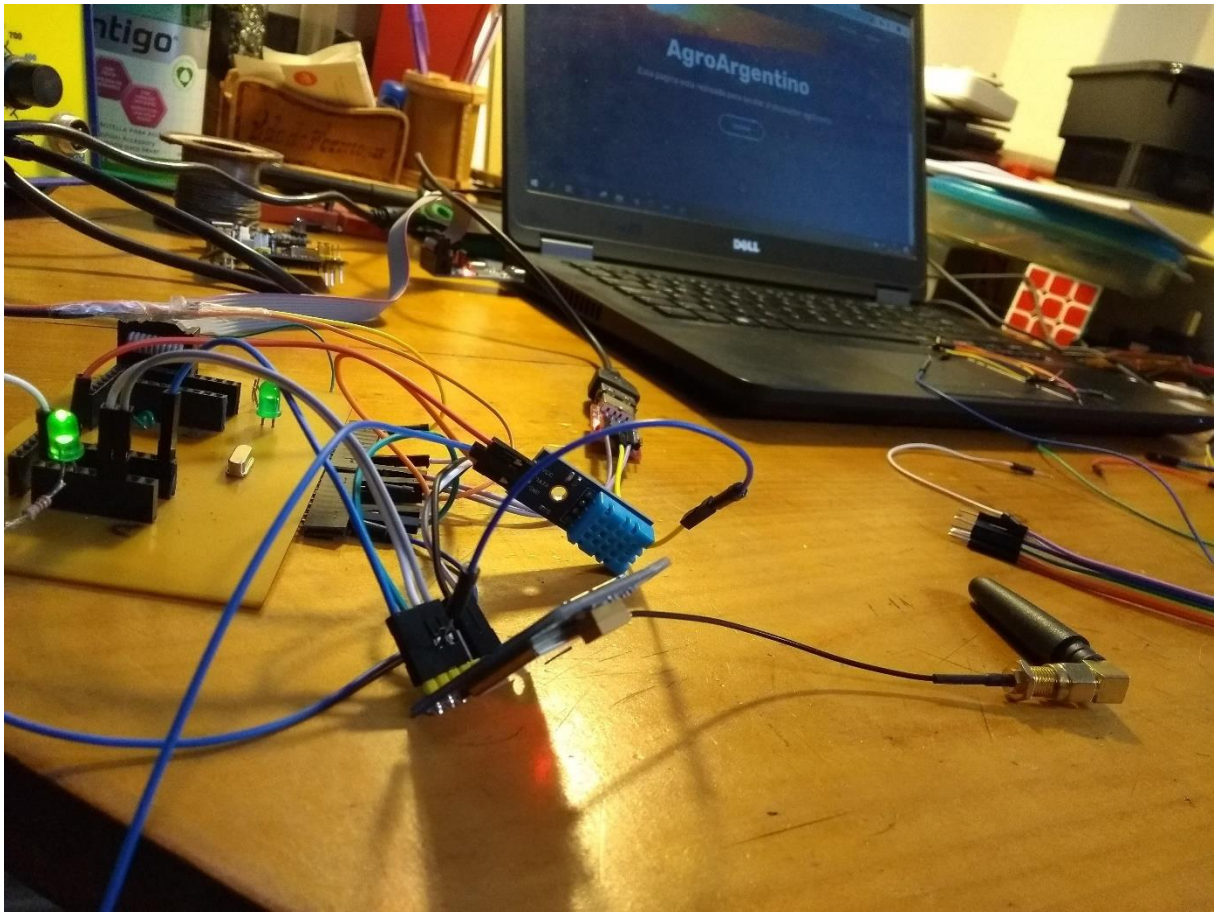
Muncial, B. (21 de 07 de 2019). *World Bank Group*.Obtenido de Banco Mundial: https://datos.bancomundial.org/indicator/AG.LND.AGRI.K2?end=2016&start=1961&view=chart

MySQL. (9 de Marzo de 2019). Obtenido de https://www.mysql.com/

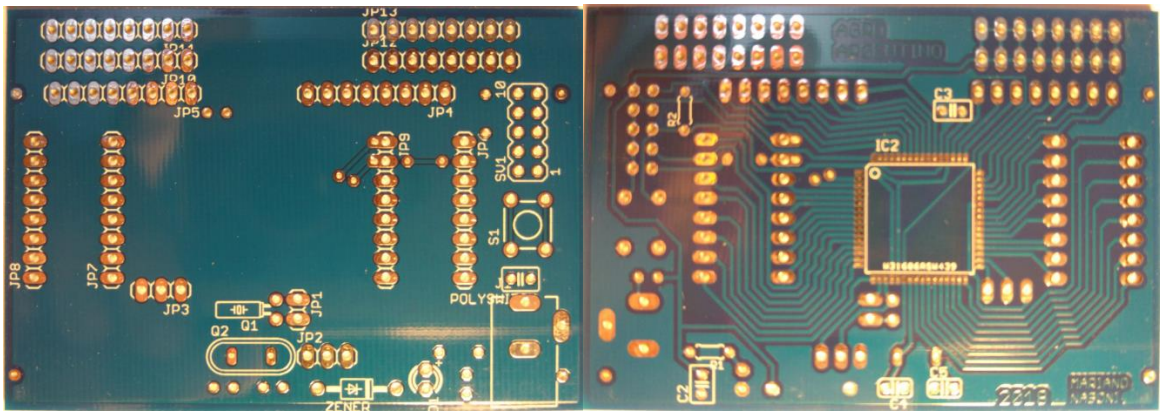
NIC. (31 de 1 de 2019). Obtenido de Registro de nombres de dominios: https://nic.ar/

13. Anexo 1

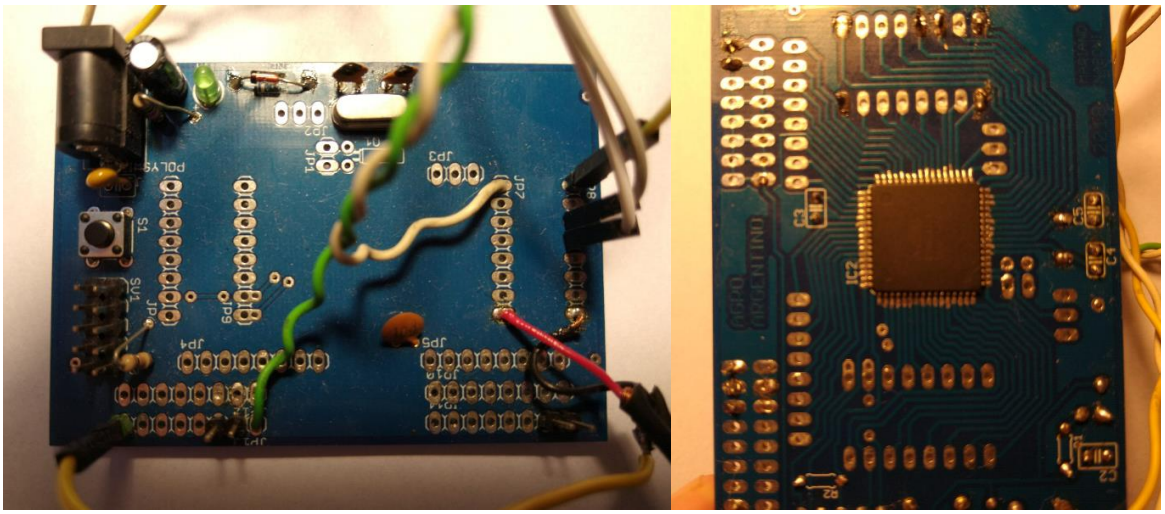
Primeras implementaciones con placa armada experimentalmente.



Mejora de la placa, terminado profesional.

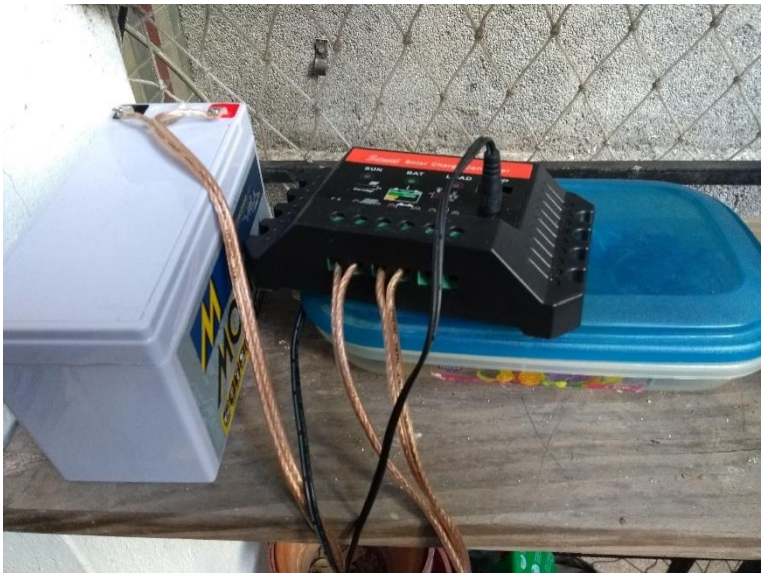


Placa con componentes soldado para eliminar falsos contactos.



Pruebas de funcionamiento de carga del panel con el sistema en funcionamiento.

Batería y cargador de batería para paneles solares.



Sistema de pivot y Panel Solar.


Etapa final de la estación.

14. Anexo 2

Datasheet básico del microcontrolador Atmega128.

Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 133 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers + Peripheral Control Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 128K Bytes of In-System Self-programmable Flash program memory
  - 4K Bytes EEPROM
  - 4K Bytes Internal SRAM
  - Write/Erase cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Up to 64K Bytes Optional External Memory Space
  - Programming Lock for Software Security
  - SPI Interface for In-System Programming
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - Two Expanded 16-bit Timer/Counters with Separate Prescaler, Compare Mode and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Two 8-bit PWM Channels
  - 6 PWM Channels with Programmable Resolution from 2 to 16 Bits
  - Output Compare Modulator
  - 8-channel, 10-bit ADC
    - 8 Single-ended Channels
    - 7 Differential Channels
    - 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Dual Programmable Serial USARTs
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
  - Software Selectable Clock Frequency
  - ATmega103 Compatibility Mode Selected by a Fuse
  - Global Pull-up Disable
- I/O and Packages
  - 53 Programmable I/O Lines
  - 64-lead TQFP and 64-pad QFN/MLF
- Operating Voltages
  - 2.7 - 5.5V for ATmega128L
  - 4.5 - 5.5V for ATmega128
- Speed Grades
  - 0 - 8 MHz for ATmega128L
  - 0 - 16 MHz for ATmega128

ATMEL

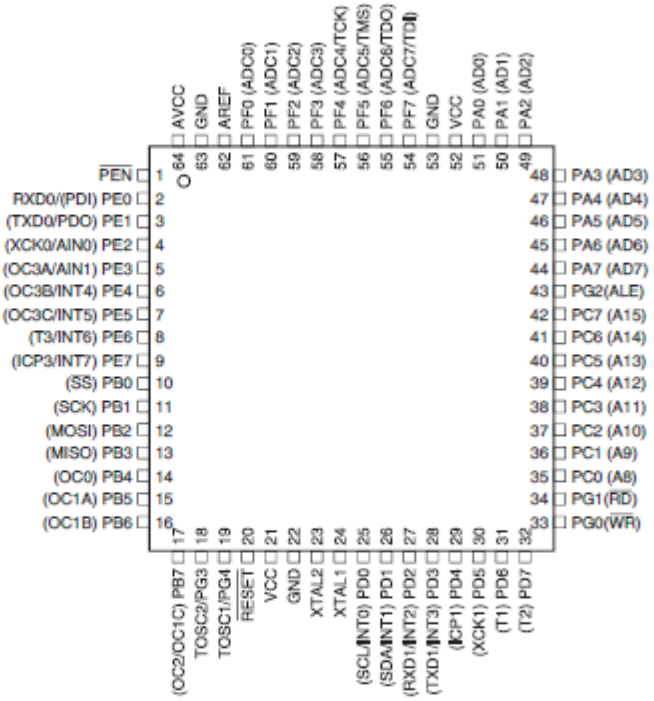
8-bit AVR® Microcontroller with 128K Bytes In-System Programmable Flash

ATmega128
ATmega128L

Rev. 2467P–AVR–08/07

Pin Configurations

Figure 1. Pinout ATmega128



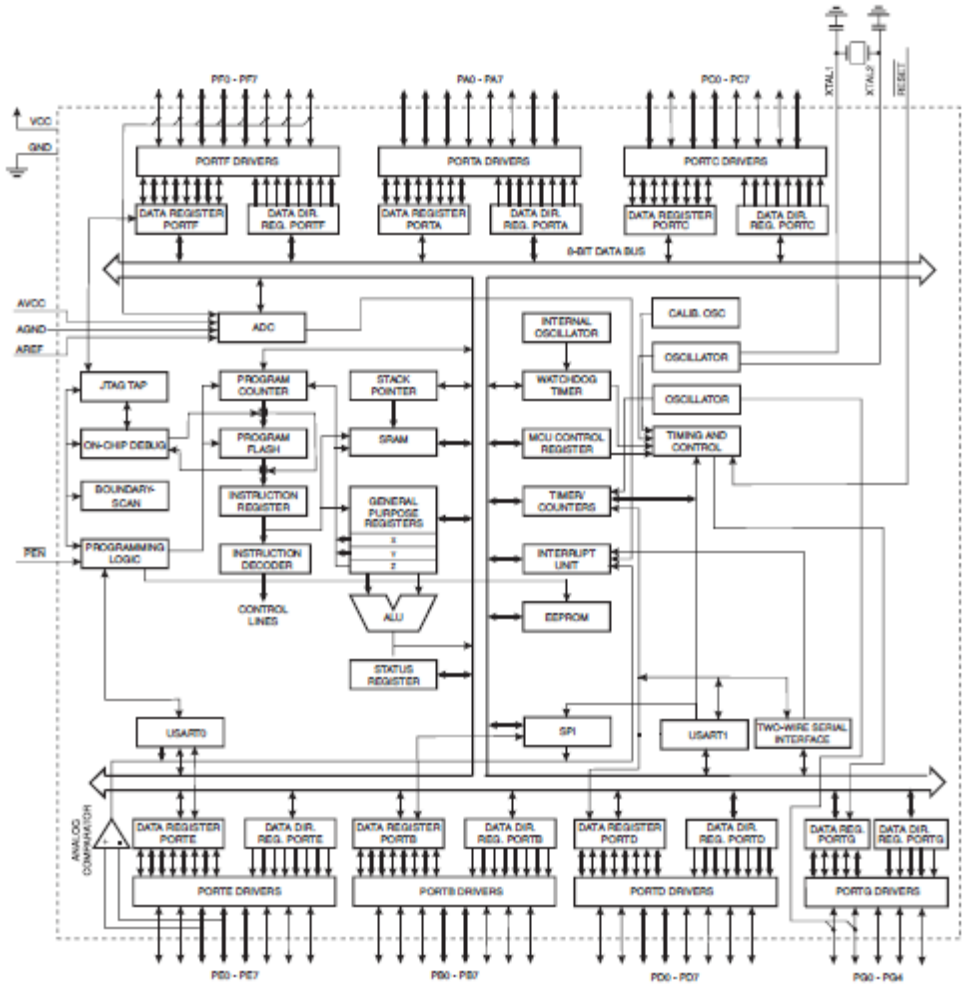
Note: The Pinout figure applies to both TQFP and MLF packages. The bottom pad under the QFN/MLF package should be soldered to ground.

Overview

The ATmega128 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega128 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 2. Block Diagram



ATmega103 Compatibility Mode

By programming the M103C fuse, the ATmega128 will be compatible with the ATmega103 regards to RAM, I/O pins and interrupt vectors as described above. However, some new features in ATmega128 are not available in this compatibility mode, these features are listed below:

- One USART instead of two, Asynchronous mode only. Only the eight least significant bits of the Baud Rate Register is available.
- One 16 bits Timer/Counter with two compare registers instead of two 16-bit Timer/Counters with three compare registers.
- Two-wire serial interface is not supported.
- Port C is output only.
- Port G serves alternate functions only (not a general I/O port).
- Port F serves as digital input only in addition to analog input to the ADC.
- Boot Loader capabilities is not supported.
- It is not possible to adjust the frequency of the internal calibrated RC Oscillator.
- The External Memory Interface can not release any Address pins for general I/O, neither configure different wait-states to different External Memory Address sections.

In addition, there are some other minor differences to make it more compatible to ATmega103:

- Only EXTRF and PORF exists in MCUCSR.
- Timed sequence not required for Watchdog Time-out change.
- External Interrupt pins 3 - 0 serve as level interrupt only.
- USART has no FIFO buffer, so data overrun comes earlier.

Unused I/O bits in ATmega103 should be written to 0 to ensure same operation in ATmega128.

Pin Descriptions

VCC

Digital supply voltage.

GND

Ground.

Port A (PA7..PA0)

Port A is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port A pins that are externally pulled low will source current if the pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port A also serves the functions of various special features of the ATmega128 as listed on page 73.

Port B (PB7..PB0)

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the ATmega128 as listed on page 74.

Port C (PC7..PC0)

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up

resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port C also serves the functions of special features of the ATmega128 as listed on page 77. In ATmega103 compatibility mode, Port C is output only, and the port C pins are **not** tri-stated when a reset condition becomes active.

Note: The ATmega128 is by default shipped in ATmega103 compatibility mode. Thus, if the parts are not programmed before they are put on the PCB, PORTC will be output during first power up, and until the ATmega103 compatibility mode is disabled.

Port D (PD7..PD0) Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATmega128 as listed on page 78.

Port E (PE7..PE0) Port E is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port E output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port E pins that are externally pulled low will source current if the pull-up resistors are activated. The Port E pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port E also serves the functions of various special features of the ATmega128 as listed on page 81.

Port F (PF7..PF0) Port F serves as the analog inputs to the A/D Converter.

Port F also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port F output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port F pins that are externally pulled low will source current if the pull-up resistors are activated. The Port F pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PF7(TDI), PF5(TMS), and PF4(TCK) will be activated even if a Reset occurs.

The TDO pin is tri-stated unless TAP states that shift out data are entered.

Port F also serves the functions of the JTAG interface.

In ATmega103 compatibility mode, Port F is an input Port only.

Port G (PG4..PG0) Port G is a 5-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port G output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port G pins that are externally pulled low will source current if the pull-up resistors are activated. The Port G pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port G also serves the functions of various special features.

The port G pins are tri-stated when a reset condition becomes active, even if the clock is not running.

In ATmega103 compatibility mode, these pins only serves as strobes signals to the external memory as well as input to the 32 kHz Oscillator, and the pins are initialized to PG0 = 1, PG1 = 1, and PG2 = 0 asynchronously when a reset condition becomes active, even if the clock is not running. PG3 and PG4 are oscillator pins.

$\overline{\text{RESET}}$ Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 19 on page 51. Shorter pulses are not guaranteed to generate a reset.

XTAL1 Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

XTAL2 Output from the inverting Oscillator amplifier.

AVCC AVCC is the supply voltage pin for Port F and the A/D Converter. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.

AREF AREF is the analog reference pin for the A/D Converter.

PEN PEN is a programming enable pin for the SPI Serial Programming mode, and is internally pulled high . By holding this pin low during a Power-on Reset, the device will enter the SPI Serial Programming mode. $\overline{\text{PEN}}$ has no function during normal operation.

Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package[1]	Operation Range
8	2.7 - 5.5V	ATmega128L-8AC ATmega128L-8MC	64A 64M1	Commercial (0°C to 70°C)
		ATmega128L-8AI ATmega128L-8AU[2] ATmega128L-8MI ATmega128L-8MU[2]	64A 64A 64M1 64M1	Industrial (-40°C to 85°C)
16	4.5 - 5.5V	ATmega128-16AC ATmega128-16MC	64A 64M1	Commercial (0°C to 70°C)
		ATmega128-16AI ATmega128-16AU[2] ATmega128-16MI ATmega128-16MU[2]	64A 64A 64M1 64M1	Industrial (-40°C to 85°C)

Notes: 1. The device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.
2. Pb-free packaging alternative, complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.

15. Anexo 3

15.1. main.c

```
#define F_CPU 1000000UL
#include<avr/io.h>
#include<string.h>
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
#include<util/delay.h>
#include<avr/interrupt.h>
#include<avr/wdt.h>
#include"uart.h"
#include"IO_Macros.h"
#include"timer1.h"
#include"Dht_22.h"
#include"wdt.h"

#define TAM_BUFFER                  300
#define TIEMPO_EXCEDIDO             2000
#define BORRADO_RESPUESTA           2

#define APN                         "wap.gprs.unifon.com.ar"
#define USERNAME                    "wap"
#define PASSWORD                    "wap"

#define DOMAIN                      "agroargentino.com.ar"
#define PORT                        "80"
#define NOMBRE                      "AGROAR001"

enum SIM_ESTADO {
        SIM_ESPERA,
        SIM_TERMINADO,
        SIM_TIEMPO_EXCEDIDO,
        SIM_RESPUESTA,
        SIM_ERROR
};

int8_t Estado_de_Respuesta, CONT_ESTADO = 0, EstadoActual=0;
uint8_t I_RH,D_RH,I_Temp,D_Temp,CheckSum;
uint32_t TimerOut = 0;

char MYSTRING_RESPUESTA[TAM_BUFFER];
char GPS[50];
char printtemp[100], printhum[100],printchecksum[100];

char conteo=0, conteo2=0;

volatile char SaltoGPS=0;
volatile uint16_t Counter = 0;
volatile uint32_t error_uno=0;
volatile int ESTADO_GPRS=1;
volatile int DarGPS=1;
```

```
int ERROR=0;
int ContReset=0;
int Pulsado=0;
int val1=0, val2=0, modo=0, botonestado=0;
int interrupcion=0;
int gprs_posicion=0;
int Minutos=10;
int contadorPivot=0;
int sumatotal=0;

float temperatura = 0;
float humedad = 0;
float checksum = 0;

void Borrar_Mystring()
{
	memset(MYSTRING_RESPUESTA,0,TAM_BUFFER);
	Counter = 0;
}

void Inicio_Espera_Respuesta()
{
	Estado_de_Respuesta = SIM_RESPUESTA;
	do {
		Leer_Respuesta();
	} while(Estado_de_Respuesta == SIM_ESPERA);

}

bool GPS_Init(){

	UART_TxString("AT+SAPBR=1,1\n\r");
	if(!EsperandoRespuesta("OK"))
		return false;

	UART_TxString("AT+SAPBR=2,1\n\r");
	if(!EsperandoRespuesta("OK"))
		return false;

	UART_TxString("AT+CIPGSMLOC=1,1\n\r");
	_delay_ms(40);
	Borrar_Mystring();
	if(!EsperandoRespuesta("OK"))
	{
		strncpy(GPS, MYSTRING_RESPUESTA+14, 34);
		return false;
	}

	UART_TxString("AT+SAPBR =0,1\n\r");
	return EsperandoRespuesta("OK");
}

bool EsperandoRespuesta(char* ExpectedResponse)
{
	Borrar_Mystring();
	_delay_ms(200);
	wdt_reset();
	Inicio_Espera_Respuesta();
```

```
if((Estado_de_Respuesta != SIM_TIEMPO_EXCEDIDO) && (strstr(MYSTRING_RESPUESTA, ExpectedResponse) != NULL))
                return true;
        return false;
}

void despertarGPRS()
{
        UART_TxString("AT\n\r");
        _delay_ms(10);
        UART_TxString("AT+CSCLK=0\n\r"); */
        _delay_ms(100);
        UART_TxString("AT\n\r");
        _delay_ms(10);
        SetBit(PORTC,7);
        _delay_ms(400);
        wdt_reset();
}

bool Envio_AT_Respuesta(char* ATCommand, char* ExpectedResponse)
{
        UART_TxString(ATCommand);
        UART_TxChar('\r');
        return EsperandoRespuesta(ExpectedResponse);
}

void ReiniciarGPRS(void)
{
        ClearBit(PORTC,RST_GPRS_PIN);
        _delay_ms(1000);
        SetBit(PORTC,RST_GPRS_PIN);
        _delay_ms(200);
        ContReset=0;
        wdt_reset();
}

void Leer_Respuesta()
{
        char CRLF_BUF[2];
        char CRLF_FOUND;
        uint32_t TimerContador = 0, LongitudRespuesta;
        while(1)
        {
                wdt_reset();
                if(TimerContador >= (TIEMPO_EXCEDIDO+TimerOut))
                {
                        CONT_ESTADO = 0; TimerOut = 0;
                        Estado_de_Respuesta = SIM_TIEMPO_EXCEDIDO;
                        return;
                }
                if(Estado_de_Respuesta == SIM_RESPUESTA)
                {
                        CRLF_FOUND = 0;
                        memset(CRLF_BUF, 0, 2);
                        Estado_de_Respuesta = SIM_ESPERA;
                }
                LongitudRespuesta = strlen(MYSTRING_RESPUESTA);
                if (LongitudRespuesta)
                {
```

```
                        _delay_ms(1);
                        TimerContador++;
                        if (LongitudRespuesta==strlen(MYSTRING_RESPUESTA))
                        {
                                for (uint16_t i=0;i<LongitudRespuesta;i++)
                                {
                                        memmove(CRLF_BUF, CRLF_BUF + 1, 1);
                                        CRLF_BUF[1] = MYSTRING_RESPUESTA[i];
                                        if(!strncmp(CRLF_BUF, "\r\n", 2))

                                        {
                                        if(++CRLF_FOUND == (BORRADO_RESPUESTA+CONT_ESTADO))
                                                {
                                                        CONT_ESTADO = 0; TimerOut = 0;
                                                        Estado_de_Respuesta = SIM_TERMINADO;
                                                        return;
                                                }
                                        }
                                }
                                CRLF_FOUND = 0;
                        }
                }
                _delay_ms(1);
                TimerContador++;
        }
}


bool GPRS_Verifico()
{
        UART_TxString("AT+CGATT=1\r");
        if(EsperandoRespuesta("OK"))
                return true;
        return false;
}

bool SIM_Inicio()
{
        for (uint8_t i=0;i<3;i++)
        {
                if(Envio_AT_Respuesta("ATE0","OK")||Envio_AT_Respuesta("AT","OK"))
                        return true;
        }
        return false;
}


bool TCP_Apagar()
{
        UART_TxString("AT+CIPSHUT\r");
        return EsperandoRespuesta("OK");
}

bool tempyhum(){
        if(dht_temperaturahumedad(&temperatura, &humedad) != -1)
         {
                dtostrf(temperatura, 3, 2, printtemp);

                dtostrf(humedad, 3, 2, printhum);
```

```
		checksum=temperatura+humedad;

		dtostrf(checksum, 3, 2, printchecksum);

		return true;
	}
	else
	{
		return false;
	}

}

bool TCP_Cerrar()
{
	UART_TxString("AT+CIPCLOSE=1\r");
	return EsperandoRespuesta("OK");
}


bool ReseteoGPRS(void)
{
	ContReset=ContReset+1;
	if(ContReset==20){
		ReiniciarGPRS();
	}
	SIM_Inicio();
	return true;
}


bool TCP_Conectar(char* _APN, char* _USERNAME, char* _PASSWORD)
{
	UART_TxString("AT+CREG?\r");
	if(!EsperandoRespuesta("+CREG: 0,1"))
	{
		ReseteoGPRS();
		return false;
	}

	UART_TxString("AT+CGATT?\r");
	if(!EsperandoRespuesta("+CGATT: 1"))
	return false;

	UART_TxString("AT+CSTT=\"");
	UART_TxString(_APN);
	UART_TxString("\",\"");
	UART_TxString(_USERNAME);
	UART_TxString("\",\"");
	UART_TxString(_PASSWORD);
	UART_TxString("\"\r");
	if(!EsperandoRespuesta("OK"))
	{
		ReseteoGPRS();
		return false;
	}

	UART_TxString("AT+CIICR\r");
```

```
	if(!EsperandoRespuesta("OK"))
	return false;

	UART_TxString("AT+CIFSR\r");
	if(!EsperandoRespuesta("."))
	return false;

	ContReset=0;

	UART_TxString("AT+CIPSPRT=1\r");
	return EsperandoRespuesta("OK");

}


uint8_t TCP_Iniciar(char* Domain, char* Port)
{
	UART_TxString("AT+CIPMUX?\r");
	if(EsperandoRespuesta("+CIPMUX: 0"))
	UART_TxString("AT+CIPSTART=\"TCP\",\"");
	else
	{
		UART_TxString("AT+CIPSTART=\"");
		UART_TxString(CANT_CONEXION);
		UART_TxString("\",\"TCP\",\"");
	}

	UART_TxString(Domain);
	UART_TxString("\",\"");
	UART_TxString(Port);
	UART_TxString("\"\r");

	CONT_ESTADO = 2;

	if(!EsperandoRespuesta("CONNECT OK"))
	{
		if(Estado_de_Respuesta == SIM_TIEMPO_EXCEDIDO)
		returnSIM_TIEMPO_EXCEDIDO;
		returnSIM_ERROR;
	}
	returnSIM_TERMINADO;
}

void Inicio_General()
{
	ClearBit(DDRD,Pivot);
	SetBit(DDRD,LED_ON);
	SetBit(PORTD,LED_ON);
	SetBit(PORTD,Pivot);

	SetBit(DDRC,RST_GPRS_PIN);
	SetBit(DDRC,DHT22_PIN);
	SetBit(PORTC,DHT22_PIN);
	SetBit(PORTC,RST_GPRS_PIN);
}

uint8_t TCPEnvio_Servidor(char* Data)
{
	UART_TxString("AT+CIPSEND\r");
```

```
	CONT_ESTADO = -1;
	EsperandoRespuesta(">");
	UART_TxString(Data);
	UART_TxString("\r\n");
	UART_TxChar(0x1A);

	if(!EsperandoRespuesta("SEND OK"))
	{
		if(Estado_de_Respuesta == SIM_TIEMPO_EXCEDIDO)
		returnSIM_TIEMPO_EXCEDIDO;
		returnSIM_ERROR;
	}
	returnSIM_TERMINADO;
}


void CONT_Init()
{
	EICRA |=(1<<ISC00);
	EIMSK |=(1<<INT0);
}

ISR(INT0_vect)
{
	if(TestBit(PIND,0))
	{
		Pulsado=1;
	}
	else
	{
		Pulsado=0;
	}
	val1=Pulsado;
	_delay_us(10);
	val2=Pulsado;
	if(val1==val2)
	{
		if(val1!=botonestado)
		{
			if(val1==1)
			{
				_delay_us(20);
				if(modo==0)
				{
					modo=1;
					contadorPivot=contadorPivot+1;
				}
				else
				{
					modo=0;
					contadorPivot=contadorPivot+1;
				}
			}
		}
		botonestado=val1;
	}
}
```

```
ISR(TIMER1_COMPA_vect)
{
	interrupcion++;
	if ( interrupcion==15*Minutos )
	 {
		SaltoGPS++;
		if(SaltoGPS==120)
		{
			SaltoGPS=0;
		}
	interrupcion = 0;
	ESTADO_GPRS=0;
	}
}

ISR (USART1_RX_vect)
{
	MYSTRING_RESPUESTA[Counter] = UDR1;
	Counter++;
	if(Counter == TAM_BUFFER)
	Counter = 0;
}

int main()
{
	char buffer_aux[300];
	unsignedchar Sample = 0;

	init_wdt();
	Inicio_General();
	CONT_Init();
	UART_Init(2400);
	timer1_init();
	sei();

	_delay_ms(20);
	contadorPivot=0;

	wdt_reset();
	ReiniciarGPRS();
	wdt_reset();
	while(1)
	{
		if(ESTADO_GPRS==0)
		{
			gprs_posicion=ESTADO_GPRS;
			ESTADO_GPRS=1;
		}
		switch(gprs_posicion)
		{
		case 0:{
			wdt_reset();
			despertarGPRS();
			if(SaltoGPS==0)
			{
				ReiniciarGPRS();
```

```
                        }
                gprs_posicion++;
                }
                break;
                case 1:{
                        while(!SIM_Inicio());
                        TCP_Apagar();
                        GPRS_Verifico();
                        while(!(TCP_Conectar(APN, USERNAME, PASSWORD)));
                        if(SaltoGPS==0)
                        {
                                GPS_Init();
                        }
                gprs_posicion++;
                }
                break;
                case 2:{
                        while(!tempyhum());
                        ClearBit(PORTC,7);
                        gprs_posicion++;
                }
                break;
                case 3:{
                        GPRS_Verifico();
                        while(!TCP_Iniciar(DOMAIN, PORT));

                        memset(buffer_aux, 0, 300);
                        sumatotal=sumatotal+contadorPivot;
                        sprintf(buffer_aux, "GET
http://agroargentino.com.ar/lluvia.php?GPS=%s&Contador=%d&TicksLluvia=%d&Humedad=%s&Temp=%s&Ch
eckSum=%s&Nombre=%s", GPS,Sample++,sumatotal,printhum,printtemp,printchecksum,NOMBRE);
                        while(!TCPEnvio_Servidor(buffer_aux));
                        _delay_ms(600);
                        TCP_Cerrar();
                        if(sumatotal>=20000)
                        {
                                sumatotal=0;
                        }
                contadorPivot=0;
                gprs_posicion++;
                }
                break;
                case 4:{
                        UART_TxString("AT+CSCLK=2\n\r");
                        gprs_posicion++;
                }
                break;
                case 5:{
                        wdt_reset();
                        _delay_ms(500);
                }
                }
        }
}
```

15.2. IO_Macros.h

```
#ifndef IO_MACROS_H_
#define IO_MACROS_H_

#define RST_GPRS_PIN 0
#define LED_ON  5
#define DHT22_PIN 7
#define Pivot      0

#define DHT_DDR DDRD
#define DHT_PORT PORTD
#define DHT_PIN PIND
#define DHT_INPUTPIN PD7
typedefstruct{
        unsignedcharB0:1;
        unsignedcharB1:1;
        unsignedcharB2:1;
        unsignedcharB3:1;
        unsignedcharB4:1;
        unsignedcharB5:1;
        unsignedcharB6:1;
        unsignedcharB7:1;
}bit_t;

#define RegBit(reg,bit) ((volatile bit_t*)&reg)->B##bit

#define RST RegBit(PORTC,0);
#define SetBit(reg,bit)      reg|=(1<<bit)
#define ClearBit(reg,bit)   reg&=~(1<<bit)
#define TglBit(reg,bit)      reg^=(1<<bit)
#define TestBit(reg,bit)     (reg&(1<<bit))


#endif/* IO_MACROS_H_ */
```

15.3. Timer.c

```
#include<avr/io.h>
#include<avr/interrupt.h>

void timer1_init()
{

  TCCR1B |= (1 << WGM12)|(0<<CS12)|(1 << CS11)|(1 << CS10);
  TCCR1A = 0;
   TCNT1 = 0;
   OCR1A=62500;
   TIMSK |= (1 << OCIE1A);


}
```

15.4. Timer.h

```
#ifndef  TIMER1_H_
#define TIMER1_H_

void timer1_init(void);

#endif/* TIMER1_H_ */
```

15.5. Uart.c

```
#include <avr/interrupt.h>
#include "uart.h"

void UART_Init(unsignedlong BAUDRATE)
{
	UCSR1B |= (1 << RXEN1) | (1 << TXEN1) | (1 << RXCIE1);
	UCSR1C |= (1 << UCSZ10) | (1 << UCSZ11);
	UBRR1L = BAUD_PRESCALE;
	UBRR1H = (BAUD_PRESCALE >> 8);
}

char UART_RxChar()
{
	while (!(UCSR1A & (1 << RXC1)));
	return(UDR1);
}

void UART_TxChar(char data)
{
	UDR1 = data;
	while (!(UCSR1A & (1<<UDRE1)));
}

void UART_TxString(char *str)
{
	int i=0;
	while (str[i]!=0)
	{
		UART_TxChar(str[i]);
		i++;
	}
}
```

15.6. Uart.h

```
#ifndef UART_H_
#define UART_H_

#define F_CPU 1000000UL
#include <avr/io.h>

#define BAUD_PRESCALE ((( F_CPU / 16) + (BAUDRATE / 2)) / (BAUDRATE )) - 1)

void UART_Init(unsignedlong);
char UART_RxChar();
void UART_TxChar(char);
void UART_TxString(char*);

#endif/* UART_H_ */
```

15.7. Wdt.c

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/wdt.h>

void init_wdt()
{
	wdt_disable();
	_delay_ms(10);
	wdt_enable(WDTO_2S);
}
```

15.8. Wdt.h

```
#ifndef WDT_H_
#define WDT_H_

void init_wdt(void);

#endif/* WDT_H_ */
```

15.9. Dht22.c

```
#include "Dht_22.h"
#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include <stdio.h>
#include <avr/wdt.h>
#include "IO_Macros.h"
```

```
int8_t dht_getdata(float *temperatura, float *humedad)
{
	uint8_t bits[5];
	uint8_t i,j = 0;

	wdt_reset();

	memset(bits, 0, sizeof(bits));

	DHT_DDR |= (1<<DHT_INPUTPIN);
	DHT_PORT |= (1<<DHT_INPUTPIN);
	_delay_ms(100);

	DHT_PORT &= ~(1<<DHT_INPUTPIN);
	_delay_us(500);
	DHT_PORT |= (1<<DHT_INPUTPIN);
	DHT_DDR &= ~(1<<DHT_INPUTPIN);
	_delay_us(40);

	if((DHT_PIN & (1<<DHT_INPUTPIN))) {
		return -1;
	}
	_delay_us(80);

	if(!(DHT_PIN & (1<<DHT_INPUTPIN))) {
		return -1;
	}
	_delay_us(80);

	uint16_t timeoutcounter = 0;
	for (j=0; j<5; j++)
	{
		uint8_t result=0;
		for(i=0; i<8; i++)
		{
			timeoutcounter = 0;
			while(!(DHT_PIN & (1<<DHT_INPUTPIN)))
			{
				timeoutcounter++;
				if(timeoutcounter > DHT_TIMEOUT)
				{
					return -1;
				}
			}
			_delay_us(30);
			if(DHT_PIN & (1<<DHT_INPUTPIN))
				result |= (1<<(7-i));
			timeoutcounter = 0;
			while(DHT_PIN & (1<<DHT_INPUTPIN))
			 {
				timeoutcounter++;
				if(timeoutcounter > DHT_TIMEOUT)
				{
					return -1;
				}
			}
		}
		bits[j] = result;
	}
```

```
	DHT_DDR |= (1<<DHT_INPUTPIN);
	DHT_PORT |= (1<<DHT_INPUTPIN);
	_delay_ms(100);

	if ((uint8_t)(bits[0] + bits[1] + bits[2] + bits[3]) == bits[4])
	{
		uint16_t convhumedad = bits[0]<<8 | bits[1];
		uint16_t convtemperatura = bits[2]<<8 | bits[3];
		if(convtemperatura & 0x8000) {
			*temperatura = (float)((convtemperatura & 0x7FFF) / 10.0) * -1.0;
		} else {
			*temperatura = (float)(convtemperatura)/10.0;
		}
		*humedad = (float)( convhumedad)/10.0;
		return 0;
	}
	return -1;
}

int8_t dht_temperatura(float *temperatura)
{
	float humedad = 0;
	return dht_getdata(temperatura, &humedad);
}

int8_t dht_humedad(float *humedad)
{
	float temperatura = 0;
	return dht_getdata(&temperatura, humedad);
}

int8_t dht_temperaturahumedad(float *temperatura, float *humedad)
{
	return dht_getdata(temperatura, humedad);
}
```

15.10. Dht.h

```
#ifndef DHT_22_H_
#define DHT_22_H_

#include <stdlib.h>
#include <stdint.h>

#define DHT_TIMEOUT 200

extern int8_t dht_temperatura(float *temperatura);
extern int8_t dht_humedad(float *humedad);
extern int8_t dht_temperaturahumedad(float *temperatura, float *humedad);

#endif/* DHT_22_H_ */
```