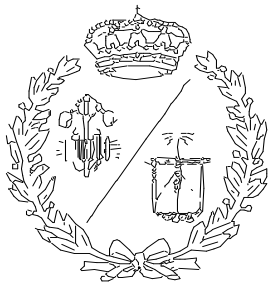
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA



Proyecto / Trabajo Fin de Carrera

Diseño y desarrollo de un sistema de monitorización y control para un invernadero de uso doméstico.

Design and development of a monitoring and control system for a domestic used Greenhouse

Para acceder al Titulo de

INGENIERO INDUSTRIAL

Autor: Cristian San Miguel Martín

Julio - 2012

Agradecimientos

Quisiera agradecer el apoyo y comprensión mostrado por mi familia, que siempre ha estado ahí cuando la he necesitado.

Asimismo, mostrar mi agradecimiento al director de este Proyecto Fin de Carrera, Pedro Corcuera Miro, por su ayuda y disponibilidad durante la realización de este proyecto.

Índice

Índice de Figuras

Índice de tablas

CAPÍTULO 1. Motivación y objetivos

1. MOTIVACIÓN

El clima juega un papel relevante en muchos aspectos de nuestra vida, uno de los más importantes es la relación directa entre el clima y nuestros cultivos.

El cambio climático puede ofrecer nuevas oportunidades de cultivos, pero también puede acarrear una reducción de las actividades agrícolas, una reducción en la calidad de los cultivos y una pérdida de rendimientos.

Además el uso cada vez más extendido de productos químicos y modificaciones transgénicas se esta produciendo que la gente opte por la instalación de invernaderos domésticos en sus propios domicilios.

La implantación de invernaderos es una solución eficaz, debida a que las condiciones en el interior de los mismo es constante, lo cual favorece el crecimiento de los cultivos, así como la intensificación de los mismos, permitiendo que funcionen durante todo el año.

Por ello toma especial importancia el desarrollo de un control de supervisión y adquisición de datos, que nos permita tener controlado tanto la luminosidad como la temperatura y humedad del suelo en el interior de los invernaderos, de una forma totalmente automatizada, todo ello con el menor coste posible, lo que nos llevaría a un ahorro tanto de agua (solo se riega cuando bajamos un % de humedad del suelo) como de electricidad al iluminar solo cuando es necesario y sobretodo de tiempo, al no tener que dedicar tiempo al control de nuestro invernadero.

2. OBJETIVOS

El objetivo de este proyecto será el diseño e desarrollo de un sistema que nos permita la adquisición de datos y control tanto automatizado como manual, de las

magnitudes temperatura, humedad del suelo y luminosidad, de un invernadero doméstico, en nuestro caso construido especialmente para este proyecto.

Este objetivo global se puede dividir en diferentes objetivos parciales del proyecto de fin de carrera, que son:

1. Conseguir medidas de los parámetros temperatura, humedad del suelo y luminosidad a partir de sensores.

2. Creación de un sistema de medición en tiempo real de los parámetros anteriormente mencionados.

3. Diseño y desarrollo del software por medio de la utilización del programa LabVIEW.

4. Control de los parámetros temperatura, humedad del suelo y luminosidad de forma automatizada y manual.

5. Visualización del muestreo de datos por medio de una interfaz de usuario con gráficos en tiempo real.

6. Almacenamiento de los datos en un fichero de texto o en Microsoft Excel, y archivarlos con su fecha y hora.

Estos objetivos se llevaran a cabo buscando un sistema de bajo coste y diseño sencillo, de fácil implantación en cualquier instalación.

CAPÍTULO 2. Introducción a los sistemas SCADA

1. SCADA

1. Introducción

Los sistemas SCADA (Supervisory Control And Data Adquisition) son aplicaciones de software, diseñadas con la finalidad de controlar y supervisar procesos a distancia. Se basan en la adquisición de datos de los procesos remotos.

Se trata de una aplicación de software, especialmente diseñada para funcionar sobre ordenadores para el control, proporcionando comunicación con los dispositivos de campo (controladores autónomos, autómatas programables, etc.) y controlando los procesos de forma automática desde una computadora. Además, envía la información generada en el proceso productivo a diversos usuarios, tanto del mismo nivel como hacia otros supervisores dentro de la empresa.

Existen diversos sistemas SCADA dependiendo del fabricante y de la finalidad del sistema, por ello a la hora de decidir cuál es el más adecuado y que tener en cuenta el cumplimiento de una serie de requisitos:

- Todo sistema debe tener arquitectura abierta, es decir, debe permitir su crecimiento y expansión, así como deben poder adecuarse a las necesidades futuras del proceso y de la planta.
- La programación e instalación no debe presentar mayor dificultad, debe contar con interfaces gráficas que muestren un esquema básico y real del proceso
- Deben permitir la adquisición de datos de todo equipo, así como la comunicación a nivel interno y externo (redes locales y de gestión)
- Deben ser programas sencillos de instalar, sin excesivas exigencias de hardware, y fáciles de utilizar, con interfaces amigables para el usuario.

2. *Historia*

Los sistemas SCADA han evolucionado a través de 3 generaciones de la siguiente manera:

- Primera generación: "monolítico": En la primera generación, la informática fue realizado por los ordenadores centrales. Las redes no existían en el momento en el cual los SCADA fueron desarrollado. Aquellos sistemas SCADA fueron sistemas independientes sin conectividad con otros sistemas. Las redes de área amplia fueron diseñados más adelante por los vendedores de RTU para comunicarse con los RTU.
- Segunda generación: "distribuida": El procesamiento se distribuye a través de varias estaciones que fueron conectados a través de LAN y comparten información en tiempo real. Cada estación era responsable de una tarea particular, con lo que el tamaño y el coste de cada estación era menor que el utilizado en una primera generación.
- Tercera generación: "en red": Debido al uso de protocolos estándar y el hecho de que muchos sistemas SCADA en red son accesibles desde Internet, los sistemas eran potencialmente vulnerables para los ciber-ataques. Por otro lado, el uso de protocolos estándar y técnicas de seguridad significa que las mejoras de seguridad estándar son aplicables a los sistemas SCADA, suponiendo que reciben mantenimiento oportuno y actualizaciones.

3. *Capacidades de SCADA*

Un software SCADA debe ser capaz de ofrecer al sistema:

- Posibilidad de crear paneles de alarma, que exigen la presencia del operador para reconocer una parada o situación de alarma, con registro de incidencias.
- Generación de datos históricos de las señale de planta, que pueden ser volcados para su proceso sobre una hoja de cálculo.
- Ejecución de programas, que modifican la ley de control, o incluso anular o modificar las tareas asociadas al autómata, bajo ciertas condiciones.
- Posibilidad de programación numérica, que permite realizar cálculos aritméticos de elevada resolución sobre la CPU del ordenador.

4. Funciones básicas del sistema

A continuación se exponen las funciones principales que ha de tener un sistema SCADA:

- Supervisión remota de instalaciones y equipos: Permite al operador conocer el estado de las instalaciones y los dispositivos que componen el sistema a controlar.
- Control remoto de instalaciones y equipos: Por medio del sistema se puede activar o desactivar los equipos remotamente de manera automática y también manual. Además es posible ajustar parámetros, valores de referencia, algoritmos de control, etc.
- Procesamiento de datos: El conjunto de datos adquiridos conforman la información que alimenta el sistema, esta información es procesada, analizada, comparada con datos de otros puntos de referencia, dando como resultado una información confiable y veraz.
- Visualización gráfica dinámica: El sistema es capaz de brindar imágenes en movimiento que representen el comportamiento del proceso, dándole al operador la impresión de estar presente dentro de un sistema real. Estos gráficos también pueden corresponder a curvas de las señales analizadas en el tiempo.
- Generación de reportes: El sistema permite generar informes con datos estadísticos del proceso en un tiempo determinado por el operador.
- Representación se señales de alarma: A través de las señales de alarma se logra alertar al operador frente a un fallo o la presencia de una condición fuera de lo aceptable. Estas señales pueden ser tanto visuales como sonoras.
- Almacenamiento de información histórica: Se cuenta con la opción de almacenar los datos adquiridos.

5. Elementos del sistema

El sistema SCADA está conformado por:

- Interfaz Hombre-Máquina: Una interfaz hombre-máquina o HMI (human-machine interfaz) es el aparato que presenta los datos del proceso a un operador humano, y a través del cual el operador controla el proceso.
  El HMI está generalmente ligado a la de base de datos del sistema SCADA y al software del programa, para ofrecer tendencias, los datos de diagnóstico y gestión de la información, tales como esquemas detallados para un sensor en particular o de la máquina en general.
- Unidad Central: Una unidad central o MTU (master terminal unit) Ejecuta las acciones de mando programadas en base a los valores actuales de las variables medidas. También se encarga del almacenamiento y procesado ordenado de los datos.
- Unidad Remota: Una unidad remota o RTU (remote terminal unit) es constituida por todo elemento que envía algún tipo de información a la unidad central. Es parte del proceso a controlar y se encuentra ubicada en la planta.
- Sistemas de comunicación: Son los conjuntos de elementos que se encargan de la trasferencia de información desde los puntos a controlar, hasta el punto donde se controla y supervisa el proceso.
- Transductores: Son los elementos encargados de hacer la transformación de una señal física a una señal eléctrica o viceversa.

6. SCADA según fabricantes

En el mercado existen una amplia variedad de SCADA funcionando. Debido a que las empresas distribuidoras de sistemas SCADA son a su vez proveedores de PLC, cada compañía realiza el SCADA que pueda comunicarse fácilmente con sus productos. A la hora de elegir un SCADA hay que tener en cuenta la compatibilidad con el sistema de adquisición de datos, y a la cantidad de variables que el programa puede leer en tiempo real.

Algunos ejemplos de SCADA y su fabricante son:

SCADA	FABRICANTE
Aimax	Desin Instruments S.A.
CUBE	Orsi España S.A.
FIX	Intellution
LabVIEW y Lookout	National Instruments
Monitor Pro	Schneider Electric
Scada InTouch	LOGITEK
SYSMAC SCS	Omron
Scatt Graph 5000	ABB
WinCC y Coros LS-B/Win	Siemens
CIRNET	CIRCUTOR S.A.
FIXDMACS	Omron-Intellution
RS-VIEW32	Rockwell
GENESIS32	Iconics

Tabla 1 SCADA según Fabricante

2. LABVIEW

1. *¿Qué es LabVIEW?*

LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench) es una plataforma de diseño de sistemas y entorno de desarrollo para un lenguaje de programación visual de National Instruments.

Es una herramienta diseñada especialmente para monitorizar, controlar, automatizar y realizar cálculos complejos de señales analógicas y digitales capturadas a través de tarjetas de adquisición de datos, puertos serie y GPIBs (Buses de Intercambio de Propósito General)

Originalmente fue realizado para Apple Macintosh en 1986, LabVIEW es normalmente utilizado para la adquisición de datos, control de instrumentación, automatización industrial y realizar cálculos complejos de señales analógicas y digitales capturadas a través de tarjetas de adquisición de datos, puertos serie y

GPIBs (Buses de Intercambio de Propósito General). Funciona en una amplia variedad de plataformas como Microsoft Windows, diferentes versiones de UNIX, Linux y Mac OS X.

2. Programación en LabVIEW

El lenguaje de programación usado en LabVIEW es conocido como G, donde G simboliza que es lenguaje gráfico. Esto significa que los programas no se escriben, sino que se dibujan, facilitando de esta manera la comprensión de los mismos.

Está basado en la programación modular, lo que permite crear tareas muy complicadas a partir de módulos o sub-módulos mucho más sencillos. Además, estos módulos pueden ser usados en otras tareas.

A cada uno de los módulos creados con LabVIEW se le llama VI (Virtual Instrument), y consta de tres componentes:

- Panel frontal: Es la interfaz de usuario. Se utiliza para interactuar con el usuario con el programa está ejecutándose. Es donde podremos observar los datos en tiempo real. En el panel frontal también se definen los controles que usaremos como entradas, así como los indicadores que será donde veamos las salidas (Ver Figura 1).
- Diagrama de bloques: Es el programa propiamente dicho, es el código fuente gráfico que define el funcionamiento del VI. Una vez construido el panel frontal, el código se desarrolla usando unas representaciones gráficas de funciones (ver Figura 2) que controlarán los objetos del panel frontal. .
- Icono y conector: El icono representa un VI dentro de otro diagrama de bloques. El conector muestra los terminales disponibles para transferir datos.

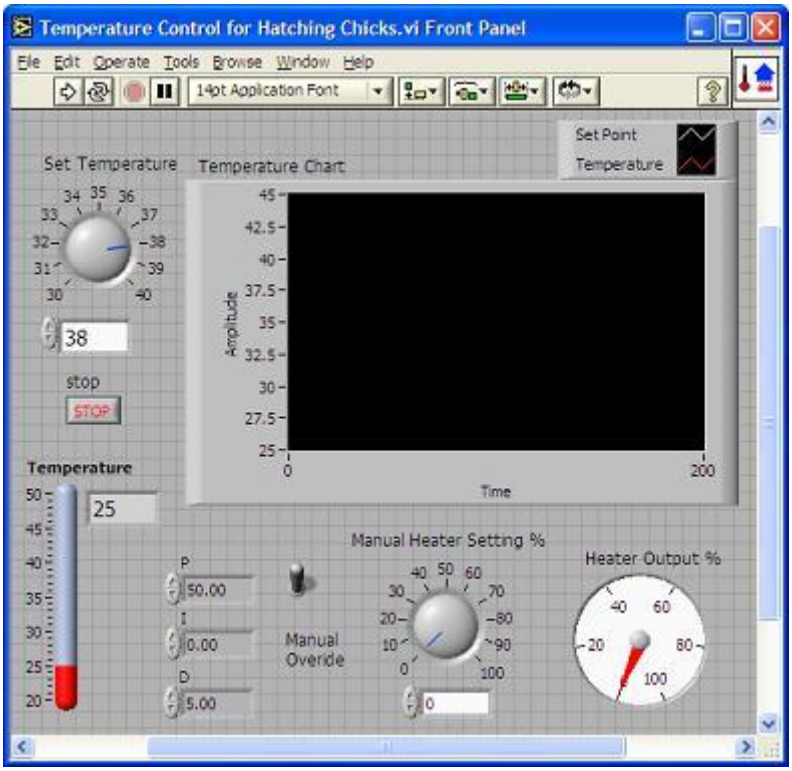


Figura 1 Panel frontal LabVIEW

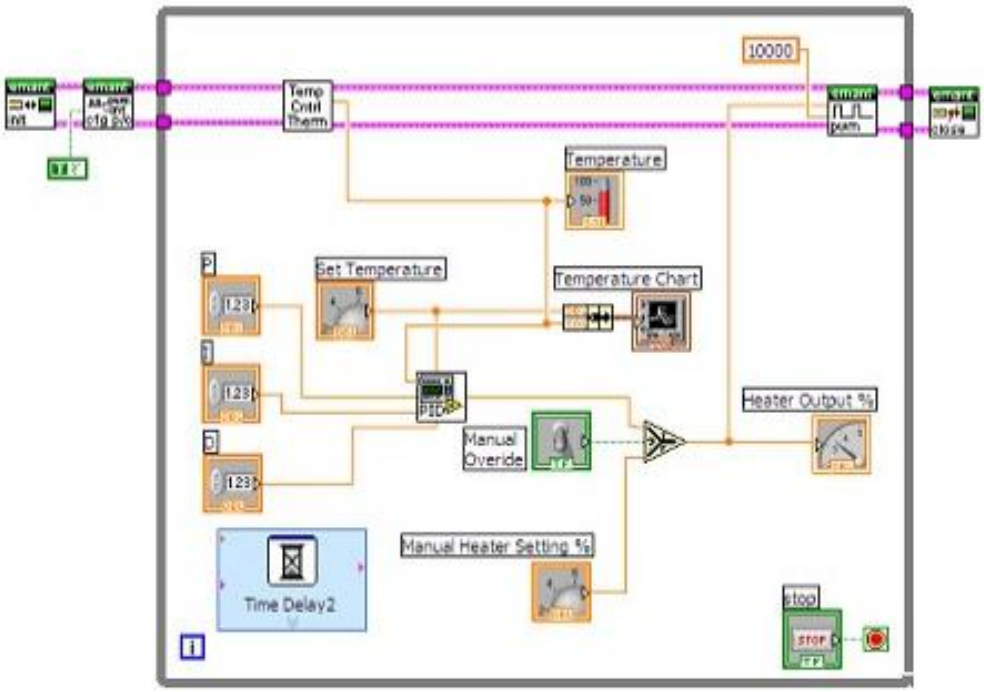


Figura 2 Diagrama de bloques LabVIEW

CAPÍTULO 3. Introducción a los invernaderos

1. ¿QUÉ ES UN INVERNADERO?

Es un invernáculo destinado a la producción de cultivos, habitualmente está cubierto exteriormente por medio de una cobertura translucida de vidrio o plástico, que permite el control de la temperatura, la humedad y otros factores ambientales, con el objetivo de favorecer el crecimiento de las plantas.

2. HISTORIA

La idea de cultivar plantas en áreas ambientalmente controladas ha existido desde los romanos. El emperador romano Tiberio se comía pepinos como su ración de verdura diaria. Los jardineros romanos usaban métodos artificiales (similar al sistema de invernadero) para tenerlo disponible para su mesa todos los días del año. Los pepinos eran cultivados debajo de telas de selenita de acuerdo con las descripciones de Plinio el viejo.

Los primeros invernaderos modernos fueron construidos en Italia en el siglo XIII después de Cristo como protección para plantas exóticas que los exploradores llevaban a Italia a su vuelta desde los trópicos. Fueron originalmente conocidos como jardines botánicos. El concepto de invernaderos pronto se extendió a los Países Bajos y luego a Inglaterra, junto con las plantas. Algunos de los primeros intentos necesitaban cantidades enormes de trabajo para mantenerlos cerrados durante la noche o el invierno. Hubo serios problemas con el suministro de calor para que este fuera adecuado y equilibrado en esos primeros invernaderos.

El botánico francés Charles Lucien Bonaparte fue quien concedió el dinero para construir el primer invernadero moderno, en la ciudad de Leiden en Holanda, para el cultivo de plantas medicinales tropicales.

Los franceses llamaron a sus primeros invernaderos orangeries, dado que los usaban para proteger los naranjos de las heladas. Cuando las piñas se hicieron populares también aparecieron los primeros invernaderos dedicados a las piñas, los cuales

llamaron pozos de piñas. La experimentación sobre el diseño de invernaderos continuo a lo largo del siglo XVII en toda Europa, así como la tecnología para producir mejores cristales y mejores técnicas constructivas. El invernadero del Palacio de Versalles fue un ejemplo por su tamaño y su grado de elaboración, al tener más de 150m de largo, 13m de ancho y 14m de alto.

En el siglo XIX se comenzó a construir invernaderos en Inglaterra. También durante este siglo se construyeron los invernaderos más grandes. Fueron concebidos tanto para horticultura como para actividades no hortícolas. Se construyeron algunos tan importantes como el Crystal Palace de Londres y New York, y el Glaspalast de Munich. Un ejemplo de arquitectura fueron los monumentales Invernaderos Reales de Laeken (1874-1895) construidos para el rey Leopoldo II de Belgica.

En el siglo XX, las cúpulas geodésicas fueron añadidas a la gran variedad de invernaderos existentes.

Las estructuras de los invernaderos fueron adaptados en la década de los 60, cuando se empezaron a cubrir con películas de polietileno. Los invernaderos de arco fueron construidos por un amplio número de compañías, así como también por los agricultores. Construidos con extrusiones de aluminio, tubos de acero galvanizado o simplemente con tubos de acero o PVC para el agua. En la década de los 70, el polietileno fue mejorados, aumentando su duración así como añadiéndole filtros UV mucho más efectivos. En la década de los 80 y los 90 se hicieron populares los invernaderos de canales conectados. Estos invernaderos tienen 2 o más bahías conectadas por una pared común, o una fila de postes de soporte. Esto produjo que tanto las entradas de calefacción como la proporción entre en suelo necesario y la superficie techada se redujera considerablemente.

Hoy en día, existen grandes superficies cubiertas por invernaderos en Holanda y en el sur de España, las cuales pueden producir millones de vegetales al año.

CAPÍTULO 4. Selección de los dispositivos

1. INTRODUCCIÓN

En este capítulo y antes de empezar el diseño se ha realizado una búsqueda de los componentes, tanto de adquisición de datos y comunicación con la computadora, como de los sensores de temperatura, humedad del suelo y luminosidad, con el objetivo de elegir correctamente los que mejor se adapten a nuestros requisitos básicos, que como se dijo en los capítulos anteriores, van a ser criterios económicos y de facilidad de instalación.

2. ADQUISICION DE DATOS Y COMUNICACIÓN

Lo primero que se estudió, fue la manera con la cual comunicaríamos los sensores con la computadora, para la adquisición de los datos, así como la conexión entre la computadora y los elementos tanto de iluminación como de ventilación y riego.

Para ello se consideraron distintos sistemas, como PLC (Programmable Logic Controller), DAQ (data acquisition modules) y Arduino.

1. DAQ

1. Introducción

Las DAQ son por lo general las interfaces entre la señal y un PC. Podría ser en forma de módulos que pueden ser conectados a la computadora de los puertos (paralelo, serie, USB, etc...) o ranuras de las tarjetas conectadas a (PCI, ISA) en la placa madre. Por lo general, el espacio en la parte posterior de una tarjeta PCI es demasiado pequeño para todas las conexiones necesarias, de modo que una ruptura de caja externa es obligatoria. El cable entre este recuadro y el PC es cara debido a los numerosos cables y el blindaje necesario y porque es exótico. Las tarjetas DAQ a menudo contienen múltiples componentes (multiplexores, ADC, DAC, TTL-IO, temporizadores de alta velocidad, memoria RAM). Estos son accesibles a través de un bus por un micro controlador, que puede ejecutar pequeños programas. El controlador es más flexible que una unidad lógica dura cableada, pero más barato

que una CPU de modo que es correcto para bloquear con simples bucles de preguntas.

2. Mercado de las DAQ

Existe una gran variedad de fabricantes de DAQ. Para el proyecto que nos concierne necesitábamos tarjetas de adquisición de datos de bajo perfil y bajo coste. De la variedad que ofrece el mercado, se podían destacar:

- Labjack U3. Precio 108€
  - 16 Flexible I/O (Entradas digitales, Salidas digitales, o entradas analógicas)
  - Hasta 2 timers.
  - Hasta 2 contadores de 32 bits
  - Hasta 16 12-bit entradas analógicas (0-2.4 V or 0-3.6 V, SE or Diff.)
  - 2 salidas analógicas (10-Bit, 0-5 volts)
  - Soporta SPI, I^2C, y Asynchronous Serial Protocols (Master Only)

Figura 3 Labjack U·

- NI USB-6501. Precio 99€
  - 24 líneas de E/S digitales
  - Un contador de 32 bits
  - Protección de sobre voltaje
  - 8.5mA de capacidad de corriente
  - Terminales de tornillo integradas o conector genérico de 34 pines (versión OEM) para fácil integración.
  - Interfaz de bus USB 2.0 de alta velocidad (12 Mb/s)

Figura 4 NI USB 6501

2. PLC

1. Introducción

Los controladores lógicos programables o PLC (Programmable Logic Controller en sus siglas en inglés) son dispositivos electrónicos muy usados en automatización industrial.

Para que un PLC logre cumplir con su función de controlar, es necesario programarlo con cierta información acerca de los procesos que se quiere secuenciar. Esta información es recibida por captadores, que gracias al programa lógico interno, logran implementarla a través de los accionadores de la instalación. Es decir, a través

de los dispositivos de entradas, formados por los sensores (transductores de entradas) se logran captar los estímulos del exterior que son procesados por la lógica digital programada para tal secuencia de proceso que a su vez envía respuestas a través de los dispositivos de salidas.

Un PLC es un equipo comúnmente utilizado en maquinarias industriales, son posibles de encontrar en todas aquellas maquinarias que necesitan controlar procesos secuenciales, así como también, en aquellas que realizan maniobras de instalación, señalización y control.

Dentro de las funciones que un PLC puede cumplir se encuentran operaciones como las de detección y de mando, en las que se elaboran y envían datos de acción a los pre-accionadores y accionadores. Además cumplen la importante función de programación, pudiendo introducir, crear y modificar las aplicaciones del programa.

2. Mercado de los PLC

Al igual que las DAQ, existe un amplio mercado de PLC. Entre los principales fabricantes de PLC podemos encontrar ABB, Allen-Bradley, Omron, Rockwell, SIEMENS, Telemecanica. El precio de los más asequibles supera los 90€.

3. ARDUINO

1. Introducción

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque (boot loader) que corre en la placa.

Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software del ordenador. Y al ser open-hardware, tanto su diseño como

su distribución son libres. Es decir, puede utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin haber adquirido ninguna licencia.

4. Selección de dispositivos

Tras observar las alternativas anteriormente expuestas, se tomó la decisión utilizar en la realización de este proyecto las placas Arduino, debido a su bajo coste, su gran flexibilidad, la posibilidad de funcionamiento autónomo (sin necesidad de conexión a pc) y la amplitud de componentes existentes, así como ser un open-hardware.

Una vez decidido que se iba a utilizar Arduino para la realización de este proyecto, pasamos a elegir la placa que más se adapte a nuestras necesidades.

A continuación se pasa a presentar las alternativas existentes en el mercado:

- Arduino Mega

  El Arduino Mega es una placa microcontrolador basada ATmeg1280 (datasheet). Tiene 54 entradas/salidas digitales (de las cuales 14 proporcionan salida PWM), 16 entradas digitales, 4 UARTS (puertos serie por hardware), un cristal oscilador de 16MHz, conexión USB, entrada de corriente, conector ICSP y botón de reset. Contiene todo lo necesario para hacer funcionar el microcontrolador; simplemente conéctalo al ordenador con el cable USB o aliméntalo con un trasformador o batería para empezar. El Mega es compatible con la mayoría de shields diseñados para el Arduino Duemilanove o Diecimila.

  - Microcontrolador ATmega1280
  - Voltaje de funcionamiento 5V
  - Voltaje de entrada (recomendado) 7-12V
  - Voltaje de entrada (limite) 6-20V
  - Pines E/S digitales 54 (14 proporcionan salida PWM)
  - Pines de entrada analógica 16
  - Intensidad por pin 40 mA
  - Intensidad en pin 3.3V 50 mA
  - Memoria Flash 128 KB de las cuales 4 KB las usa el gestor de arranque(bootloader)

- SRAM 8 KB
- EEPROM 4 KB
- Velocidad de reloj 16 MHz

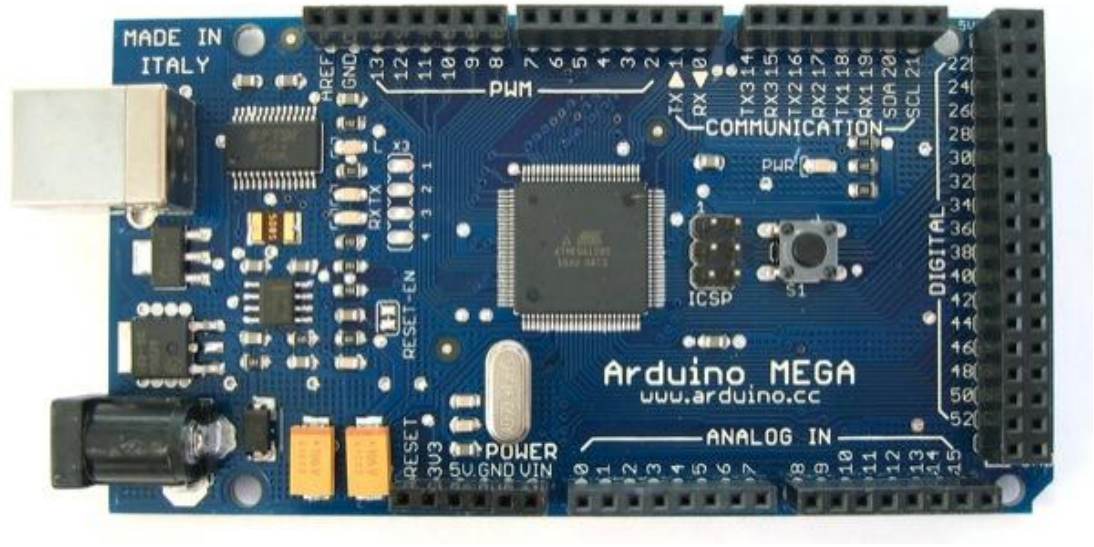


Figura 5 Arduino Mega

El Arduino Mega (ver Figura 5) puede ser alimentado vía la conexión USB o con una fuente de alimentación externa. El origen de la alimentación se selecciona automáticamente.

Las fuentes de alimentación externas (no-USB) pueden ser tanto un transformador o una batería. El transformador se puede conectar usando un conector macho de 2.1mm con centro positivo en el conector hembra de la placa. Los cables de la batería pueden conectarse a los pines Gnd y Vin en los conectores de alimentación (POWER).

La placa puede trabajar con una alimentación externa de entre 6 a 20 voltios. Si el voltaje suministrado es inferior a 7V el pin de 5V puede proporcionar menos de 5 Voltios y la placa puede volverse inestable, si se usan más de 12V los reguladores de voltaje se pueden sobrecalentar y dañar la placa. El rango recomendado es de 7 a 12 voltios.

Los pines de alimentación son los siguientes:

- VIN. La entrada de voltaje a la placa Arduino cando se está usando una fuente externa de alimentación (en opuesto a los 5 voltios de la conexión USB). Se puede proporcionar voltaje a través de este pin, o, si se está alimentado a través de la conexión de 2.1mm, acceder a ella a través de este pin.
- 5V. La fuente de voltaje estabilizado usado para alimentar el microcontrolador y otros componentes de la placa. Esta puede provenir de VIN a través de un regulador integrado en la placa, o proporcionada directamente por el USB u otra fuente estabilizada de 5V.
- 3V3. Una fuente de voltaje a 3.3 voltios generada en el chip FTDI integrado en la placa. La corriente máxima soportada 50mA.

- Arduino UNO.

El Arduino UNO es una placa microcontrolador basada ATmeg328 (datasheet). Tiene 14 entradas/salidas digitales (de las cuales 6 proporcionan salida PWM), 6 entradas analógicas, un cristal oscilador a 16Mhz, conexión USB, entrada de alimentación, una cabecera ISCP, y un botón de reset. Contiene todo lo necesario para utilizar el microcontrolador; simplemente conéctalo a tu ordenador a través del cable USB o aliméntalo con un transformador o una batería para empezar a trabajar con él.

- Microcontrolador ATmega368
- Voltaje de funcionamiento 5V
- Voltaje de entrada (recomendado) 7-12V
- Voltaje de entrada (limite) 6-20V
- Pines E/S digitales 14 (6 proporcionan salida PWM)
- Pines de entrada analógica 6
- Intensidad por pin 40 mA
- Intensidad en pin 3.3V 50 mA
- Memoria Flash 32 KB (ATmega328) de las cuales 2 KB las usa el gestor de arranque(bootloader)

- SRAM 2 KB
- EEPROM 1 KB
- Velocidad de reloj 16MHz

Figura 6 Arduino UNO

El Arduino UNO (ver Figura 6) puede ser alimentado vía la conexión USB o con una fuente de alimentación externa. El origen de la alimentación se selecciona automáticamente.

Las fuentes de alimentación externas (no-USB) pueden ser tanto un transformador o una batería. El transformador se puede conectar usando un conector macho de 2.1mm con centro positivo en el conector hembra de la placa. Los cables de la batería pueden conectarse a los pines Gnd y Vin en los conectores de alimentación (POWER).

La placa puede trabajar con una alimentación externa de entre 6 a 20 voltios. Si el voltaje suministrado es inferior a 7V el pin de 5V puede proporcionar menos de 5 Voltios y la placa puede volverse inestable, si se usan más de 12V los reguladores de voltaje se pueden sobrecalentar y dañar la placa. El rango recomendado es de 7 a 12 voltios.

Los pines de alimentación son los siguientes:

- VIN. La entrada de voltaje a la placa Arduino cando se está usando una fuente externa de alimentación (en opuesto a los 5 voltios de la conexión USB). Se puede proporcionar voltaje a través de este pin, o, si se está alimentado a través de la conexión de 2.1mm, acceder a ella a través de este pin.
- 5V. La fuente de voltaje estabilizado usado para alimentar el microcontrolador y otros componentes de la placa. Esta puede provenir de VIN a través de un regulador integrado en la placa, o proporcionada directamente por el USB u otra fuente estabilizada de 5V.
- 3V3. Una fuente de voltaje a 3.3 voltios generada en el chip FTDI integrado en la placa. La corriente máxima soportada 50mA.
- GND. Pines de toma de tierra.

Existen otras placas arduino como la Diecimila, Nano, Lilypad, Fio, Mini… pero tienen unas aplicaciones diferentes a las que nos interesan, como son Lilypad y Fio, versiones antiguas como Diecimila o versiones reducidas como Mini o Nano.

Para este proyecto, la placa elegida es la Arduino UNO en su versión rev3 que es la última. Esto se puede justificar debido a que las prestaciones de la Arduino UNO son suficientes para nuestro proyecto y debido a su menor coste, será la seleccionada. En proyectos de mayor entidad y con mayor número de sensores, se podrá utilizar la Arduino Mega.

3. SENSORES

En este apartado se van a exponer diferentes tipos de transductores que se utilizan en los instrumentos de medida y se hace especial mención en los tipos de dispositivos que son objeto en este proyecto, temperatura, luminosidad, y humedad del suelo.

1. Introducción

Un sensor es un dispositivo que detecta o mide magnitudes físicas, químicas o biológicas, llamadas variables de instrumentación, que pueden ser por ejemplo velocidad, aceleración, temperatura, luminosidad, etc., y se encarga de transformarlas en otras magnitudes fácilmente medible. Los sensores pueden de

indicación directa (como el termómetro de mercurio, que aprovecha la propiedad que posee el mercurio de dilatarse o contraerse por la acción de la temperatura) o pueden estar conectados a un indicador (convertidor analógico-digital, un ordenador…) de forma que los valores puedan ser leído por un operador y/o almacenados de forma digital.

2. Sensores fotoeléctricos

Un sensor fotoeléctrico es un dispositivo electrónico que responde al cambio de intensidad de la luz. Los sensores de luz se usan para detectar el nivel de luz y producir una señal de salida representativa respecto a la cantidad de luz detectada.

Los sensores fotoeléctricos o sensores de luz, pueden estar basados en distintas tecnologías. A continuación veremos las alternativas disponibles:

- Fotodiodo: Un fotodiodo (ver Figura 7) es un semiconductor construido con una unión PN, sensible a la incidencia de la luz visible o infrarroja. Para que su funcionamiento sea correcto se polariza inversamente, con lo que se producirá una cierta circulación de corriente cuando sea excitado por la luz.

  Ventajas: El fotodiodo responde a los cambios de oscuridad a iluminación y viceversa con mucha más velocidad, y puede utilizarse en circuitos con tiempo de respuesta más pequeño

  Suelen ser utilizados en lectores de CD.

Figura 7 Fotodiodo

- Fotorresistencia: Una fotorresistencia (ver Figura 8) es un componente electrónico cuya resistencia disminuye con el aumento de intensidad de luz incidente. El valor de resistencia eléctrica de un LDR es bajo cuando hay luz incidiendo en él (puede descender hasta 50 ohm) y muy alto cuando está a oscuras (varios megaohmios).

  La variación del valor de la resistencia tiene cierto retardo, diferente si se pasa de oscuro a iluminado o de iluminado a oscuro. Esto limita a no usar los LDR en aplicaciones en las que la señal luminosa varía con rapidez. El tiempo de respuesta típico de un LDR está en el orden de una décima de segundo

  Es utilizado en una amplitud de operaciones que no requieran tiempos de reacción del orden de la décima de segundo. Como en sensores para identificar noche y día.

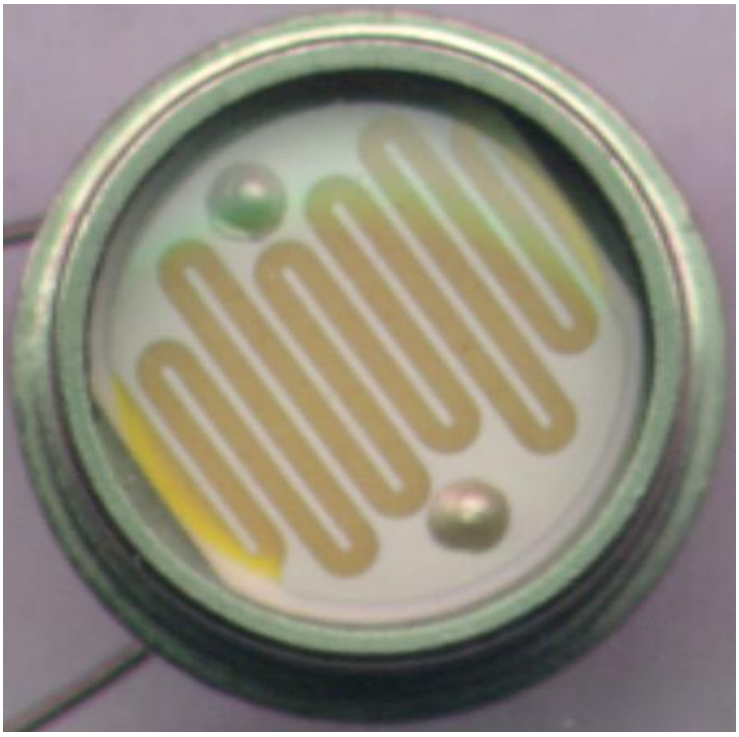


Figura 8 Fotorresistencia

- Fototransistor: La luz incide sobre la región de base, generando portadores en ella. Esta carga de base lleva el transistor al estado de conducción. El fototransistor es más sensible que el fotodiodo por el efecto de ganancia propio del transistor.

  Un fototransistor puede trabajar como un transistor común con la corriente de base y/o como un fototransistor propiamente dicho cuando la luz que incide este hace la función de la corriente base.

  Se suelen utilizar en lectores de cinta y tarjetas perforadas.

- Célula fotoeléctrica: Es un dispositivo electrónico que transforma la energía luminosa en energía eléctrica. Su uso principal es la generación eléctrica, aunque también se pueden utilizar como sensor de luz.

  Su coste no las hace óptimas para la detección de luz, por lo que suelen usarse únicamente como generadores eléctricos.

3. *Sensores de temperatura*

Un sensor de temperatura es un dispositivo capaz de transformar la variación de temperatura en una señal eléctrica capaz de ser leída por una computadora. Existen distintos tipos de sensores de temperatura. A continuación van a ser expuestas las principales características de cada tipo.

- Termopar: Basan su funcionamiento en la unión de dos metales distintos que producen un voltaje por efecto Seebeck (ver Figura 9) en función de la diferencia de temperatura que exista entre los dos extremos, los cuales se denominan “punto caliente” y “punto frio”.

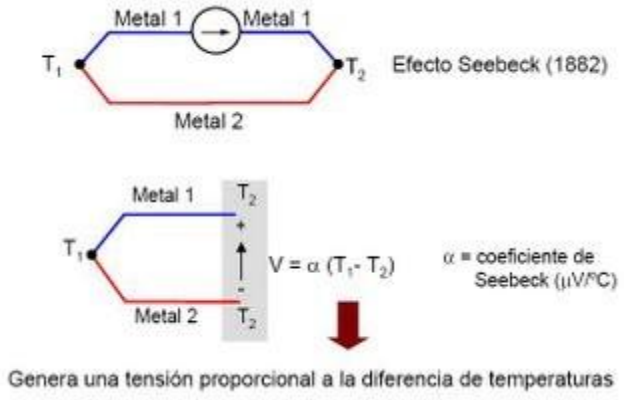


Figura 9 Termopar y Efecto Seebeck

Sus principales ventajas son su precio económico, que son intercambiables, que sus conectores son estándar y su capacidad para medir un amplio rango de temperaturas, lo que les hace ampliamente utilizados en la industria. Por el contrario su principal limitación es la exactitud, ya que difícilmente son capaces de obtener errores del sistema inferiores a un grado Celsius.

- RTD: Los RTD (Resistive Temperature Detector) son sensores de temperatura cuyo principio físico se basa en la resistividad de los metales, es decir, en variación de la resistencia de un conductor con la temperatura. Esto se debe a que al incrementar la temperatura los iones vibran con mayor amplitud y así se dificulta el paso de los electrones a través del conductor.

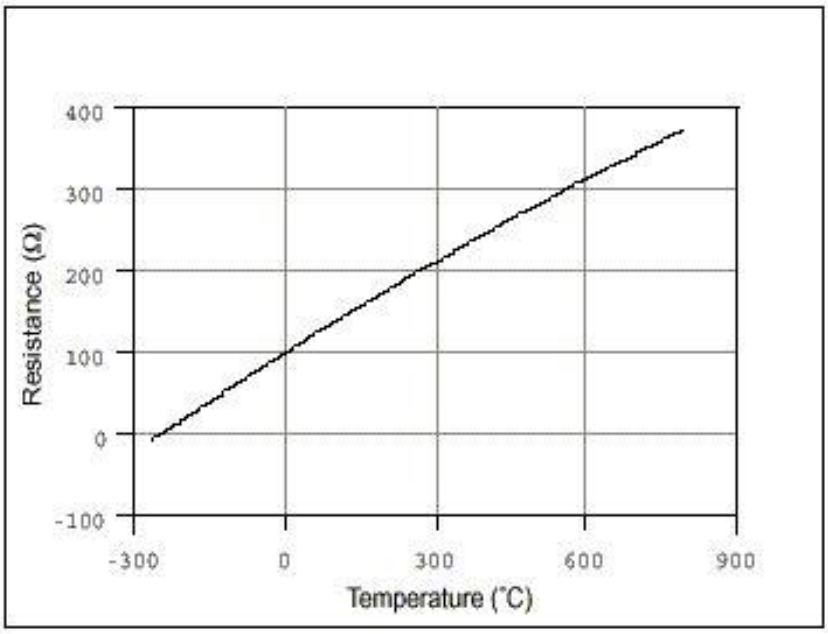


Figura 10 Curva Resistencia-Temperatura RTD de Platino 100Ω

Las características que deben tener los metales son un alto coeficiente de resistencia y alta resistividad para que tenga mayor sensibilidad y que haya una relación lineal entre la resistencia y la temperatura (ver Figura 10). La máxima calidad de los RTD la dan los detectores de platino ya que permiten realizar medidas más exactas y estables hasta una temperatura de aproximadamente 500 °C. Los RTD más económicos utilizan el níquel o aleaciones de níquel pero éstos no son tan estables ni lineales como los que emplean platino.

Sus principales ventajas son su elevada linealidad, elevada precisión, buena estabilidad a larga plazo (0.02% después de 1000h), son elementos pasivos, tiempos de respuesta pequeños, tienen elevados margen de temperatura. Por el contrario sus principales puntos negativos son su elevado coste, su poca robustez, sensible autocalentamiento y sensibilidad menor que los termistores.

- Termistor: Los termistores son semiconductores electrónicos que son sensibles a la temperatura. Existen dos tipos de termistores, los NTC (Negative Temperature Coefficient) y los PTC (Positive Temperature Coefficient). Amos tipos presentan una respuesta no lineal y decreciente con

el aumento de la temperatura en el caso de los NTC y creciente en el caso de los PTC.

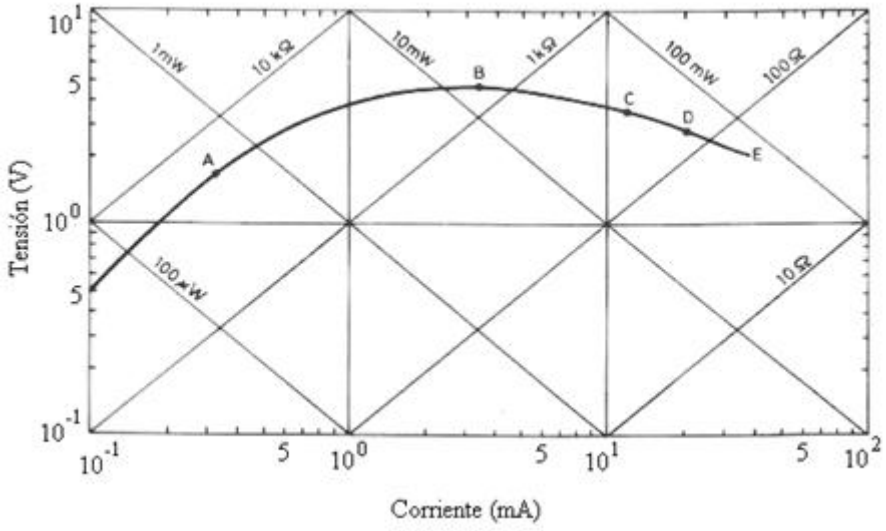


Figura 11 Tensión-corriente de un termistor NTC

Las principales ventajas son que presentan una mayor sensibilidad, inferior tiempo de respuesta, dispositivos de dimensión muy reducidas, así como un precio más económico que los RTD. Como principal desventaja es que su falta de linealidad (Ver Figura 11).

4. Sensores de humedad del suelo

Existen tres tipos de tecnología para medir la humedad del suelo: sensores de dominio de frecuencia (son de capacitancia o sensor de impedancia eléctrica), medidor de humedad de neutrones y sensores que miden la resistencia del suelo. A continuación se van a exponer las principales características de cada tipo.

- Dominio de frecuencia (FD): Es un instrumento desarrollado para medir el contenido de humedad en suelo. El instrumento tiene un circuito oscilante, la parte de detección del sensor está incrustado en el suelo, y la frecuencia de operación dependerá del valor de la constante dieléctrica del suelo.

  Dentro de los FD, existen dos tipos: Las sondas de capacitancia y los sensores de impedancia eléctrica.

Las sondas de capacitancia o sensor de capacitancia de franja son unas sondas que utilizan la capacitancia para medir la permitividad eléctrica del suelo. El volumen de agua en el volumen total de suelo tiene una fuerte influencia en la permitividad dieléctrica del suelo, porque la constante dieléctrica del agua (78.5) es muy superior a los demás constituyentes del suelo (materia orgánica: 4, suelo mineral: 4, aire: 1). Gracias a esto, cuando la cantidad de agua cambia en el suelo, la sonda puede medir el cambio en la capacitancia (por medio del cambio de la permitividad dieléctrica) que puede ser directamente correlacionada con el cambio en la cantidad de agua.

Los sensores de impedancia eléctrica se basan en la medición de la impedancia eléctrica por medio de sondas. La configuración más común se basa en el principio de onda estacionaria (Gaskin & Miller, 1996). El dispositivo comprende un oscilador sinusoidal de 100 MHz, una línea de transmisión fija de impedancia coaxial, y cables sonda enterrados en el suelo. La señal del oscilador se propaga a lo largo de las líneas de transmisión en la sonda del suelo, y si la impedancia de la sonda difiere de la de la línea de transmisión, una proporción de la señal incidente es reflejada de vuelta a lo largo de las líneas hacia la fuente de la señal. Midiendo esto, se puede obtener la cantidad de agua en suelo.

Estos sensores tienen el problema principal de necesitar ser calibrados en función del suelo a medir. Por contra, su principal ventaja es su bajo coste y respuesta rápida.

- Medidor de humedad de neutrones: Estos sensores utilizan la dispersión de los neutrones para medir la humedad. Es una técnica no destructiva y su principal ventaja es que mide la humedad en la mayor parte del suelo, no solo en las cercanías del sensor.

  El agua, debido a su contenido en hidrogeno, es un eficaz moderador de neutrones, retardando a los neutrones de alta energía. Con una fuente de electrones de alta energía y un detector sensible a los neutrones de baja

energía (neutrones térmicos), la tasa de detección estará gobernada por el contenido de agua en suelo comprendido entre la fuente y el detector.

La principal desventaja de estos sensores es su coste, no tan económico como las otras alternativas presentes.

- Sensores de resistencia del suelo: La principal ventaja de estos sensores es su sencillez y su bajo coste, se emplean habitualmente en usos domésticos. Se basan en dos electrodos introducidos en el suelo que miden la resistencia del suelo entre los dos. En función de la resistencia del suelo se puede obtener la cantidad de agua presente en él.

5. Elección de los sensores

Una vez expuestos los distintos tipos de sensores tanto de temperatura, como de humedad del suelo, así como de luminosidad, se va pasar a seleccionar los sensores a utilizar en la realización de este proyecto. Como ya he dicho en anteriores capítulos, se seguirán unos criterios de bajo coste y sencillez que hagan ampliamente utilizable este sistema.

1. Sensor de luminosidad

Una vez vistas las distintas alternativas (fotodiodo, fotorresistencia, fototransistor y célula fotoeléctrica) vamos a pasar a seleccionar el más adecuado para nuestra instalación.

En nuestro caso, su aplicación va ser la detección de noche y día. Para ello, según lo anteriormente visto, el método más adecuado es la fotorresistencia, debido a su bajo coste y su sencillez.

También debemos tener en cuenta el sistema de comunicación y control utilizado en el proyecto (arduino) el cual tiene unas limitaciones, como sus voltaje de salida, que es de 5V a intensidad por pin es de 40mA o de 3.3V a intensidad de 50mA, pero

también unas ventajas, como módulos de bajo coste y alta sencillez, especialmente diseñado para arduino.

Se puede construir un sensor de luz por medio de una fotorresistencia, un amplificador operacional, una pequeña placa y un conecto. A continuación vemos el precio de fabricación que tendría este sensor:

- FOTO-RESISTENCIA LDR 3.4 mm 0.45€
- Amplificador operacional LM358N 0.72€
- Placa Prototipos + conector 4 pines 2.45€
- El precio total del sensor seria de: 3.62€.

Por otro lado, podríamos encontrar sensores de luminosidad preconstruidos. Dentro de la gama conectar y listo especialmente diseñada para arduino, se encuentra el sensor de luz analógico preconstruido cuyo coste es de 4.25€.

Para la realización de este proyecto se ha elegido el **Modulo Sensor de Luz Conectar y Listo Analógico**, considerando que pese al aumento del precio de 0.63€, el estar preconstruido lo hace utilizable por usuarios sin conocimientos de electrónica.

2. Sensor de temperatura

Como se ha visto en puntos anteriores, existen distintos tipos de sensores de temperatura (termopares, termistores y RTD).

El sensor de temperatura escogido es del tipo resistivo ya que como se ha comentado anteriormente los termopares presentan un acondicionamiento complejo, y los RTD tienen un coste superior al buscado en este proyecto.

A continuación vamos a presentar los sensores encontrados en el mercado que pueden satisfacer nuestras necesidades:

- EB - Temperature sensor: El sensor cuenta con un termistor NTC TTC0. Su rango de temperaturas es de -40 a 125ºC con una precisión comprendida entre el 1% y el 3%. Su precio es de 2.5€.
- Grove - Temperature Sensor conectar y listo: El sensor cuenta con un termistor NTC. Su rango de temperaturas es de -40 a 125ºC con una precisión del +/-1.5ºC. Su precio es de 2.4€.
- Sensor de temperatura KTY81-222: Este sensor cuenta con un termistor PTC. Su rango de temperaturas es de -55 a 150ºC con una precisión del +/-3%. Su precio es de 1.83€.

Una vez vistos los sensores de temperatura. Se ha tomado la decisión de utilizar el Grove - Temperature Sensor, debido a que las exigencias de nuestra instalación no requieren una gran precisión, y el Grove - Temperature Sensor una alternativa económica, al mismo tiempo que la más sencilla, al contar con el conector "conectar y listo" especialmente diseñado para arduino.

3. Sensor Humedad del Suelo

Como se presentó en puntos anteriores, existen principalmente 3 tipos de sensores de humedad en suelo (FD, Medida de humedad por neutrones, Sensor de resistencia del suelo). A continuación se pasa a exponer la elección realizada en este proyecto.

Siguiendo el criterio de todo el proyecto, el tipo de sensor elegido será el de resistencia del suelo, debido a que es el de menor coste, y que además, no necesita ser calibrado en función del suelo a medir.

Debido a que existe una baja oferta de sensores de humedad en suelo, y que la mayoría de esta oferta consta de su propio medidor, y no cuentan con salida para medición externa, se ha elegido el Módulo Sensor Humedad del Suelo Conectar y Listo, que debido a su bajo coste (7.25€) y su facilidad de instalación (forma parte de la gama conectar y listo para arduino), es un sensor adecuado a este proyecto.

6. *Descripción sensores elegidos*

A continuación se va pasar a exponer detalladamente las características de los sensores elegidos.

1. Modulo Sensor de Luz Conectar y Listo Analógico

Es un sensor de la gama conectar y listo (ver Figura 12) diseñada específicamente para arduino. Este módulo consta de una fotorresistencia GL5528 para detectar la intensidad de la luz en el ambiente. Un amplificador operacional LM358 configurado como seguidor de tensión y un conector de 4 pines.

Figura 12 Modulo Sensor de Luz Conectar y Listo Analógico

Sus características electrónicas son:

Artículos	Condiciones	Mínimo	Tipo	Máximo	Unidad
Características del sistema					
VCC	-	3	5	30	V
Corriente de suministro	-	0.5	-	3	mA
Características fotorresistor					
Resistencia a la luz	10Lux	8	-	20	KW
Resistencia oscuro	0Lux	-	1	-	KW
100γ10	-	-	0.6	-	-
Tiempo de respuesta	Creciente	-	20	-	S
	Que cae	-	30	-	S
Longitud de onda máxima	-	-	540	-	Nm
La temperatura ambiente	-	-30	-	70	ºC

Tabla 2 Características del Módulo sensor de luz

A continuación se mostraran el esquema de Modulo (ver Figura 13) y la curva de resistencia de la fotorresistencia GL5528 (ver Figura 14).

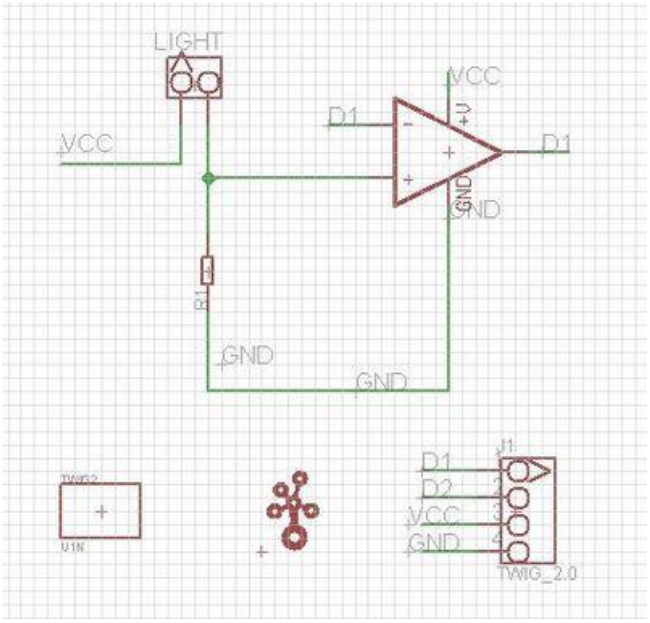


Figura 13 Esquema Modulo Sensor de Luz Conectar y Listo Analógico

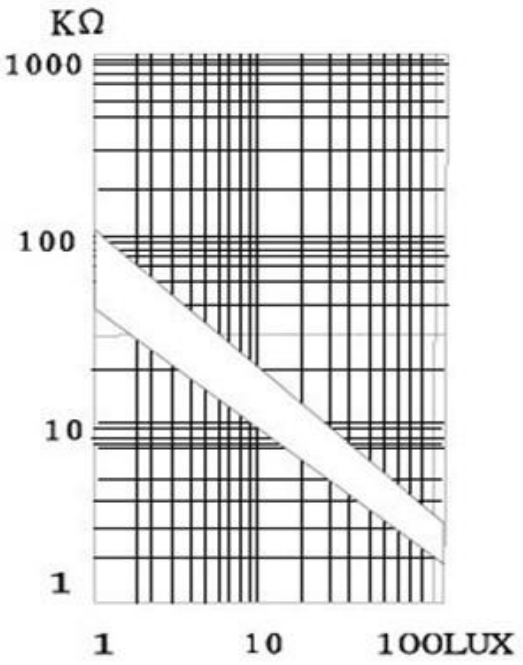


Figura 14 Curva de resistencia de la fotorresistencia GL5528

2. Módulo Sensor de Temperatura conectar y listo

Es un sensor de la gama conectar y listo (ver Figura 15) diseñada específicamente para arduino. Este módulo usa un termistor NTC TTC3A103_34D, que retorna la temperatura ambiente por medio de la variación de la resistencia que luego se utiliza para alterar Vcc (5V en nuestro arduino). Nuestra placa luego convierte ese valor del voltaje medido por el puerto analógico en temperatura. Su rango de operación es de -40 a 125ºC con una precisión de +/-1.5%.

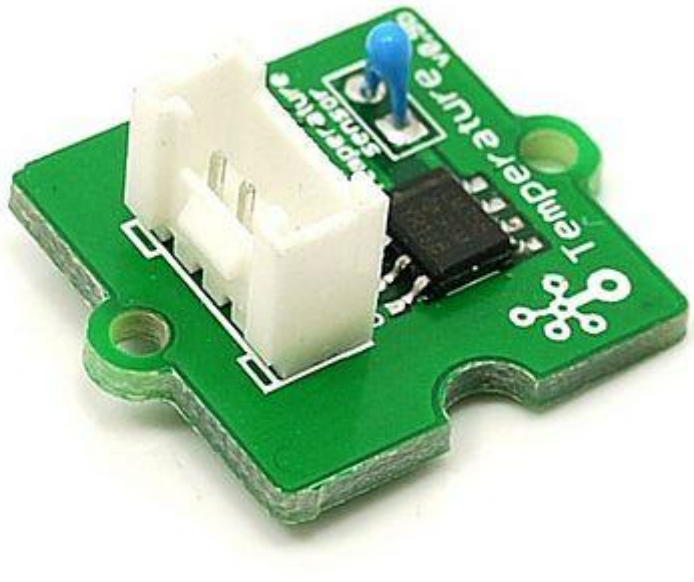


Figura 15 Grove Temperature Sensor conectar y listo

A continuación se mostraran el esquema de Modulo (ver Figura 16) y la curva de resistencia del termistor TTC3A103_34D (ver Figura 17).

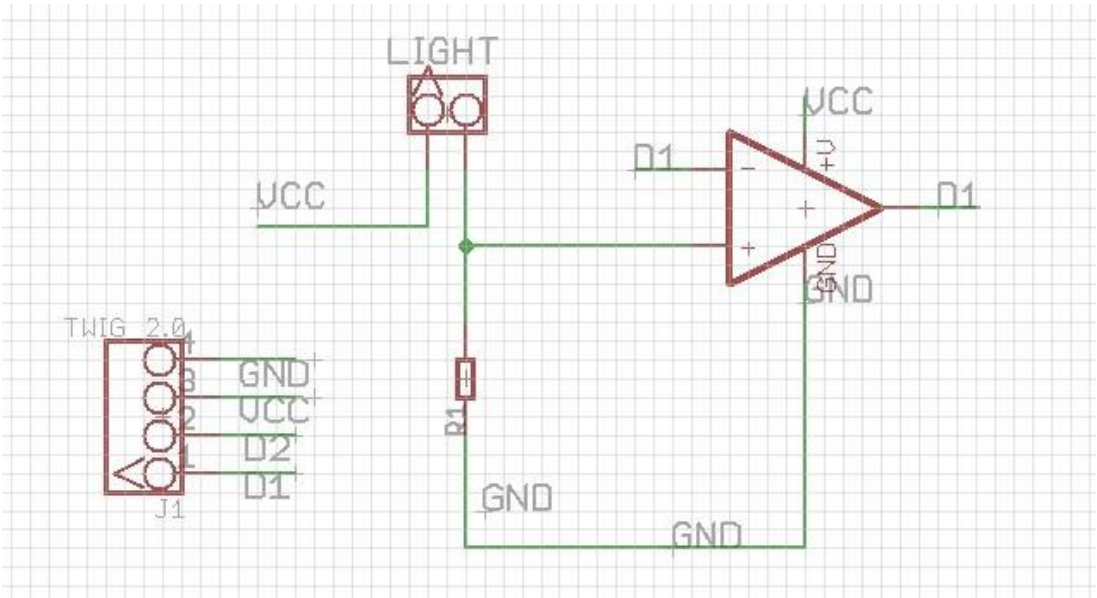


Figura 16 Esquema Grove Temperature Sensor

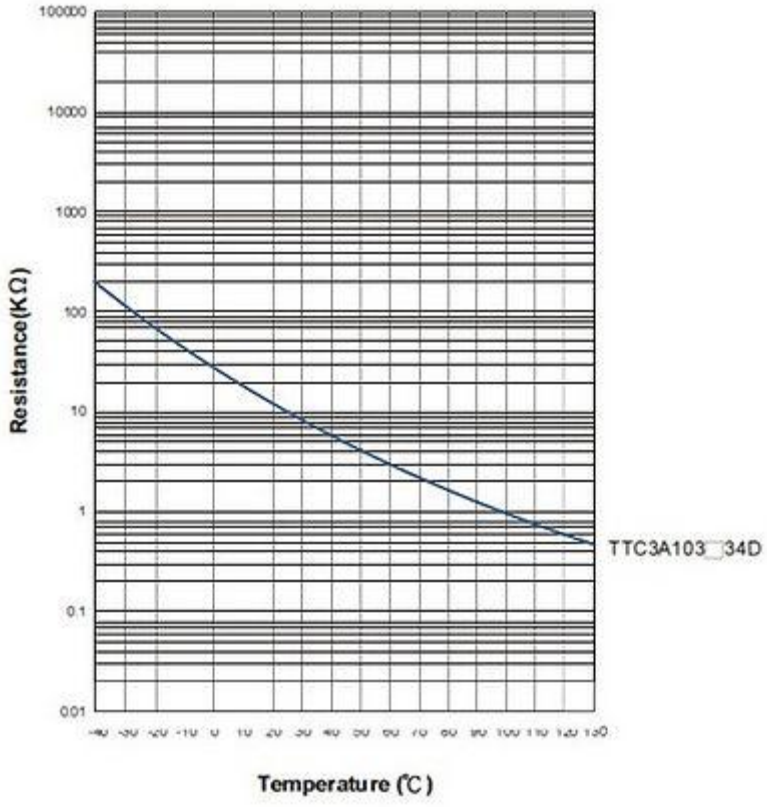


Figura 17 Curva Resistencia termistor TTC3A103_34D

A continuación se muestran las especificaciones del sensor:

Artículo	Min	Típico	Max	Unidad
Dimensiones	2.0 x 2.0			Cm
Voltaje	3.3	5	30	Voltios
Corriente	0.5	-	5	mA

Tabla 3 Especificaciones Grove Temperature Sensor

Modelo	Zero power resistance	Tolerancia de la resistencia	Máxima potencia	Disipación térmica	Constante térmica en tiempo
	R25ºC(KΩ)	+/-%	mW	mW/ºC	sec
TTC3A103_34D	10	3	150	>2.5	<18

Tabla 4 Especificaciones Termistor TTC3A103_34D

3. Módulo Sensor Humedad del Suelo Conectar y Listo

Este sensor (ver Figura 18) mide la humedad del suelo por medio de la variación de resistencia que presenta este en función de su cantidad de agua.

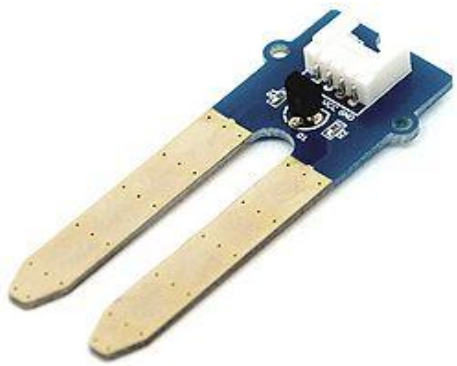


Figura 18 Módulo Sensor Humedad del Suelo

A continuación se muestran las especificaciones del sensor:

Artículo	Min	Típico	Max	Unidad
Dimensiones	2.0 x 6.0			Cm
Voltaje	3.3	-	5	Voltios
Corriente	0	-	.5	mA

Tabla 5 Especificaciones Módulo Sensor Humedad del Suelo

A continuación se mostraran tanto es esquema (ver Figura 19) como sus dimensiones mecánicas detalladas (ver Figura 20)

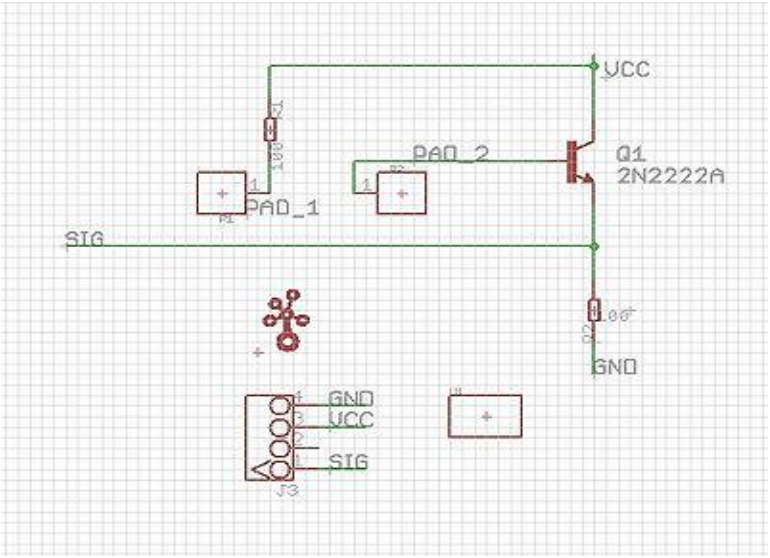


Figura 19 Esquema del Módulo Sensor Humedad del Suelo

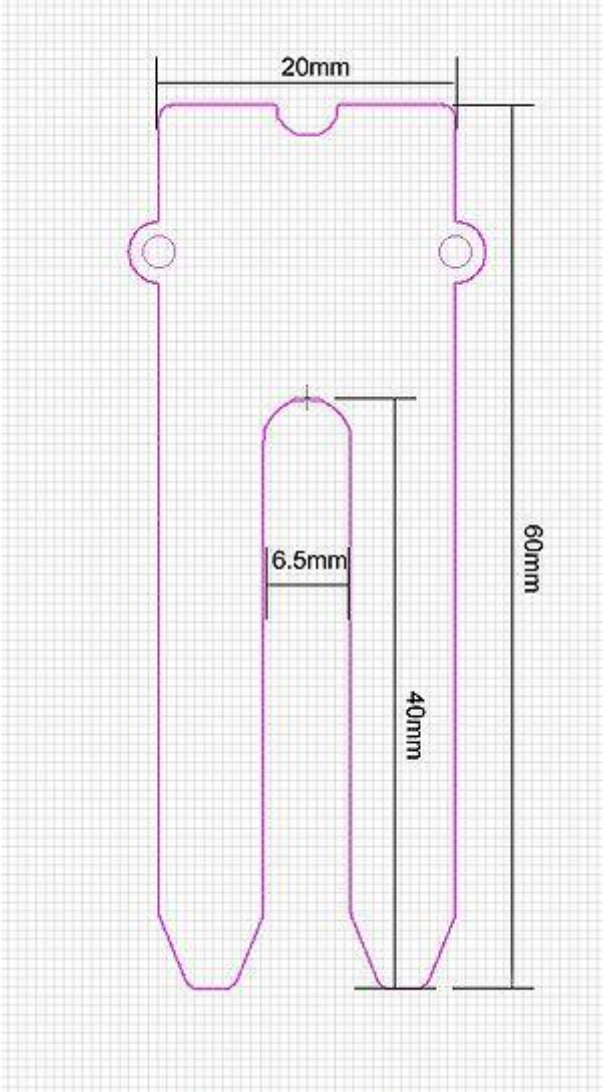


Figura 20 Dimensiones mecánicas del Módulo Sensor Humedad del Suelo

7. *Elementos adicionales*

Además de los sensores, es necesario la utilización de otros componentes para el control y la adquisición de datos.

Para el control se van a utilizar 4 relés, mientras que para el control automatizado de la iluminación será necesaria la utilización de un reloj, así como será necesaria la utilización del Arduino Shield conectar y listo (Ver Figura 21).

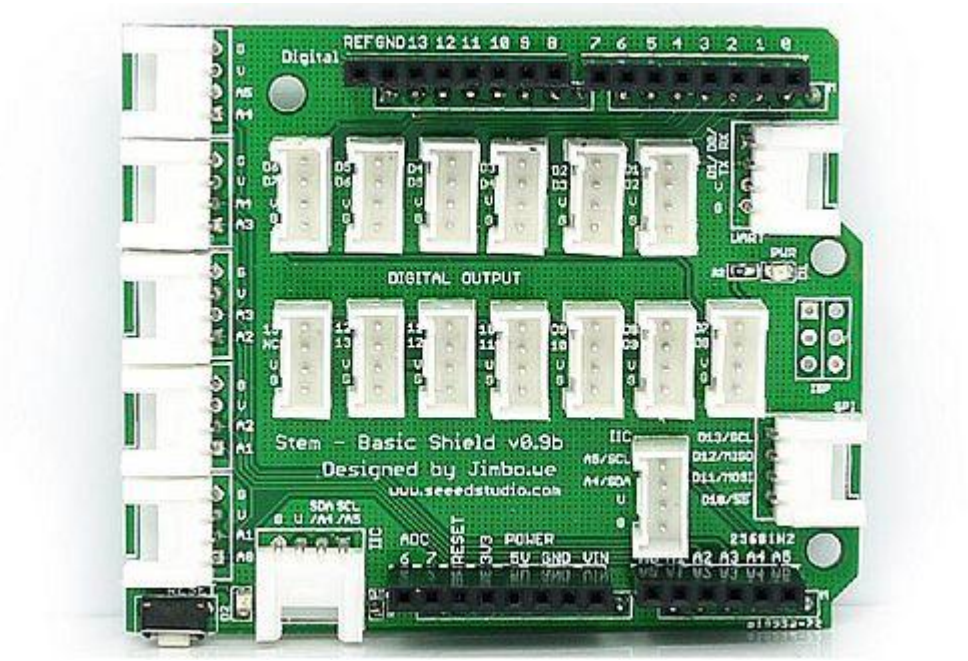


Figura 21 Arduino Shield conectar y listo

1. Elección de los relés.

En el caso de los relés se van a utilizar los modelos de la gama conectar y listo (Ver Figura 22), dado su bajo coste y su facilidad de utilización y control.

Figura 22 Módulo Relé 5V Conectar y Listo

Los valores máximos soportados por el relé son:

- 10A 250 VAC
- 15A 120 VAC
- 10A 24 VDC

2. Elección del reloj

La elección del reloj, se ha hecho siguiendo la idea de este proyecto, sencillez y bajo coste, y para ello se ha elegido el reloj tiempo real DS1307 conectar y listo (Ver Figura 23).

Figura 23 Modulo Reloj tiempo real DS1307 conectar y listo

El módulo de RTC es un miembro de Grove. Se basa en el chip DS1307 que soporta el protocolo I^2C. Utiliza una pila de litio (CR1225). El reloj / calendario proporciona segundos, minutos, horas, día, fecha, mes y año de información. El fin de mes se ajusta automáticamente para los meses con menos de 31 días, incluyendo las correcciones de año bisiesto. El reloj funciona tanto en el formato de 24 horas o 12 horas con indicador AM / PM.

Sus características son:

- El reloj de tiempo real cuenta segundos, minutos, horas, fecha del mes, mes, día de la semana y año con la compensación del año bisiesto válido hasta 2100
- 56-Byte, respaldada por baterías, no volátil (NV) de RAM para almacenamiento de datos

- I^2C interfaz serire
- 5V de alimentación en CC
- Señal de onda cuadrada programable
- Consume menos de 500nA en el modo de reserva de la batería con el oscilador en funcionamiento.

Su esquema se muestra a continuación:

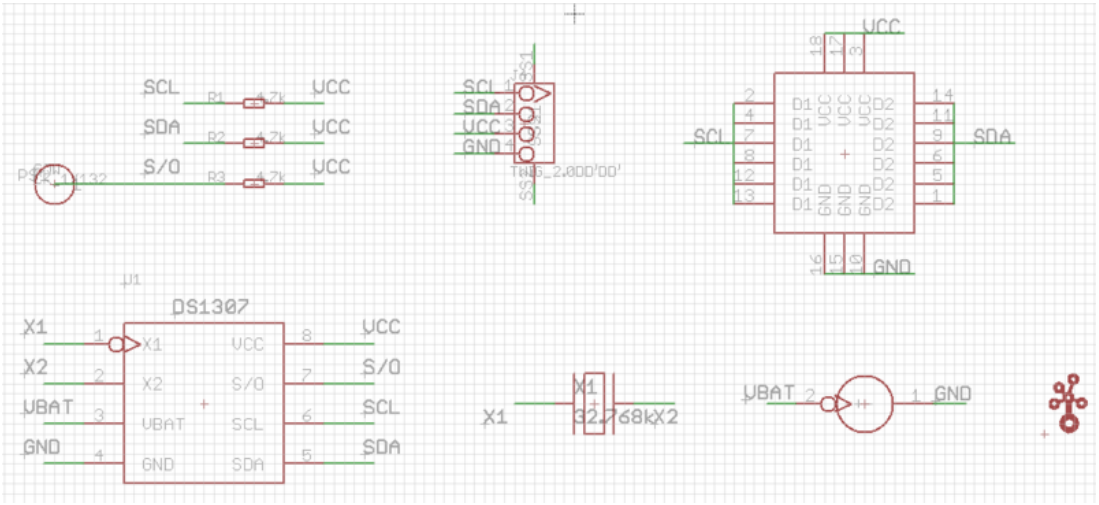


Figura 24 Esquema Modulo reloj tiempo real DS1307 conectar y listo

Sus especificaciones físicas y electrónicas son:

Artículos	Mínimo
Tamaño de la placa	2.0cm *4.0cm
Estructura IO	SCL,SDA,VCC,GND
ROHS	Si

Tabla 6 Especificaciones físicas Modulo Reloj tiempo real DS1307 conectar y listo

Artículos	Condiciones	Min	Norma	Max	Unidad
VCC	-	4.5	5.0	5.5	V
Entrada Lógica 1	-	2.2	-	VCC+0.3	V
Entrada Lógica 0	-	-0.3	-	+0.8	V
Tensión de la batería	-	2.0	3.0	3.5	V
Corriente de la batería	(OSC ON); SQW/OUT OFF	-	300	500	nA
	(OSC ON); SQW/OUT ON (32kHz)	-	480	800	nA
	Corriente de retención de datos	-	10	100	nA

Tabla 7 Especificaciones electrónicas Modulo Reloj tiempo real DS1307 conectar y listo

CAPÍTULO 5. Desarrollo del software

1. Introducción

En este capítulo se desarrolla el software del proyecto. Se divide en dos partes:

- Un primer software desarrollado completamente en arduino, el cual se utilizara para el funcionamiento sin conexión a pc de forma automatizada.
- Un segundo software desarrollado mediante el programa "LabVIEW", basado en entorno de programación gráfica, el cual se utilizara para el funcionamiento conectado a pc.

2. Programación en Arduino

1. Introducción

El programa que se utiliza para programar en arduino, es el Arduino 1.0, es un entorno de programación de código abierto, que hace fácil escribir el código y cargarlo a la placa. Funciona en Windows, Mac OS X y Linux. El entorno está escrito en Java y basado en Processing, avr-gcc y otros programas también de código abierto.

El lenguaje utilizado es Arduino, está basado en C/C++ y soporta todas las construcciones de C estándar y algunas funcionalidades de C++. Vincula la librería AVR Libc y permite el uso de todas sus funciones.

2. Conexión placa Arduino

El primer paso ha de ser conectar la placa arduino al ordenador por medio del puerto USB. Una vez conectado la placa base y el ordenador debemos instalar los drivers.
A continuación se muestran los pasos a seguir:

- Abrir el administrador de dispositivos de Windows. Para ello hacemos: Equipo/propiedades/Administrado de dispositivos:

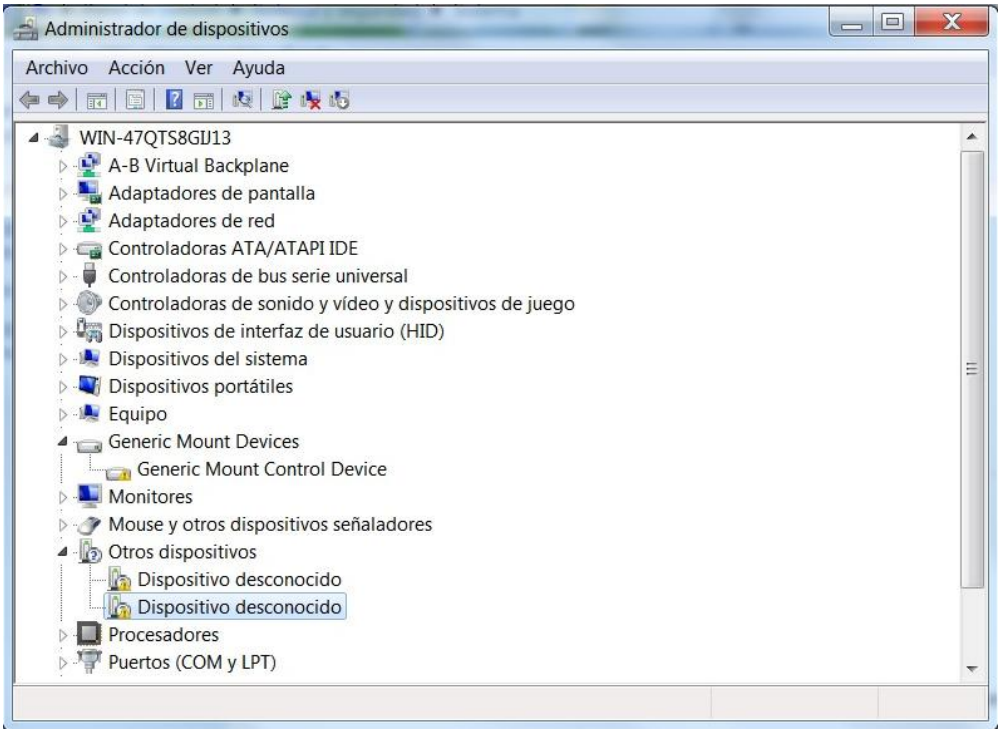


Figura 25 Administrador de dispositivos

- Como se puede observar en la Figura 25 hay dispositivos desconocidos. Abrimos el último de los dispositivos desconocidos.

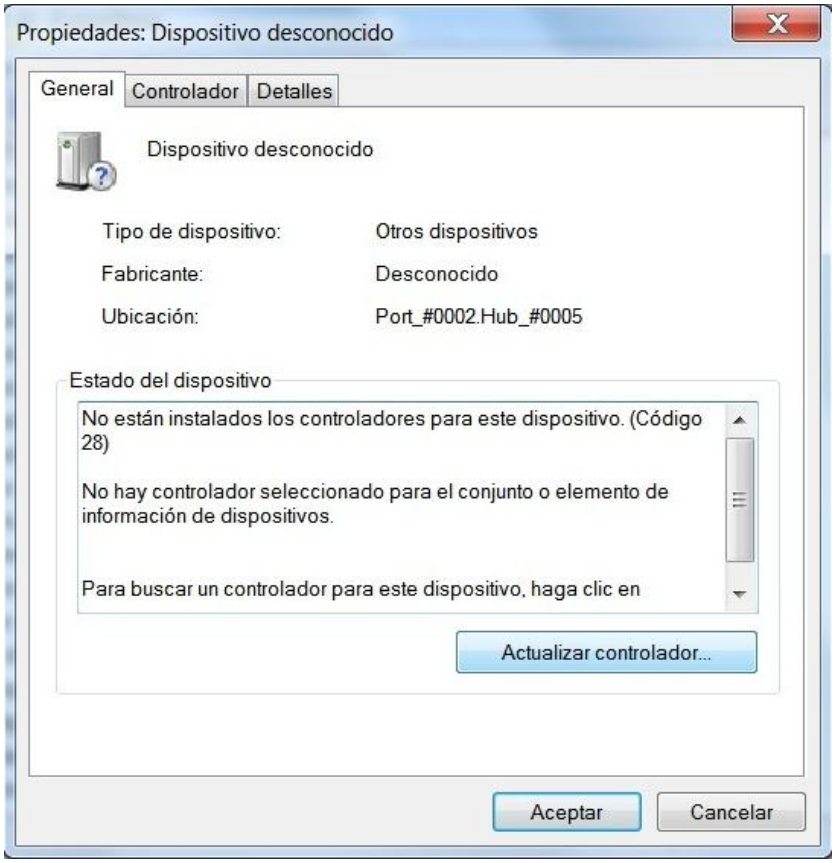


Figura 26 Propiedades Dispositivo desconocido

- Seleccionamos el botón actualizar controlador y a continuación seleccionamos la opción buscar software de controlador de equipo (ver Figura 27).

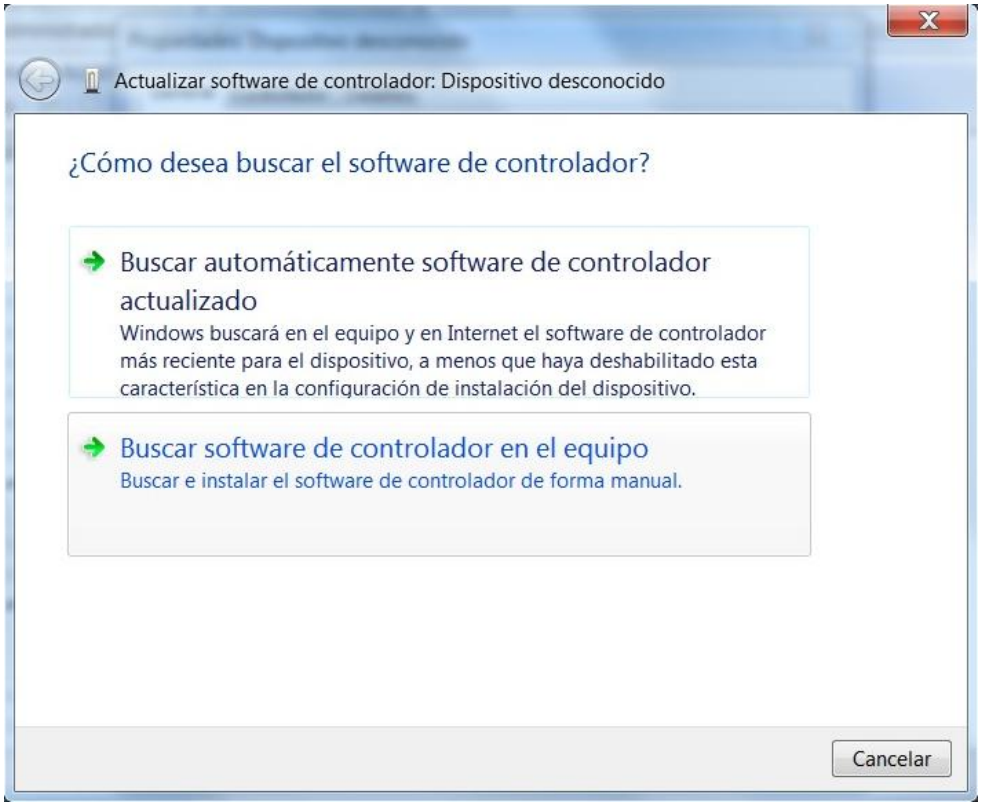


Figura 27 Actualizar software de controlador

- Escribimos la ruta donde tenemos instalado el programa Arduino 1.0 y marcamos la opción incluir subcarpetas (ver Figura 28)

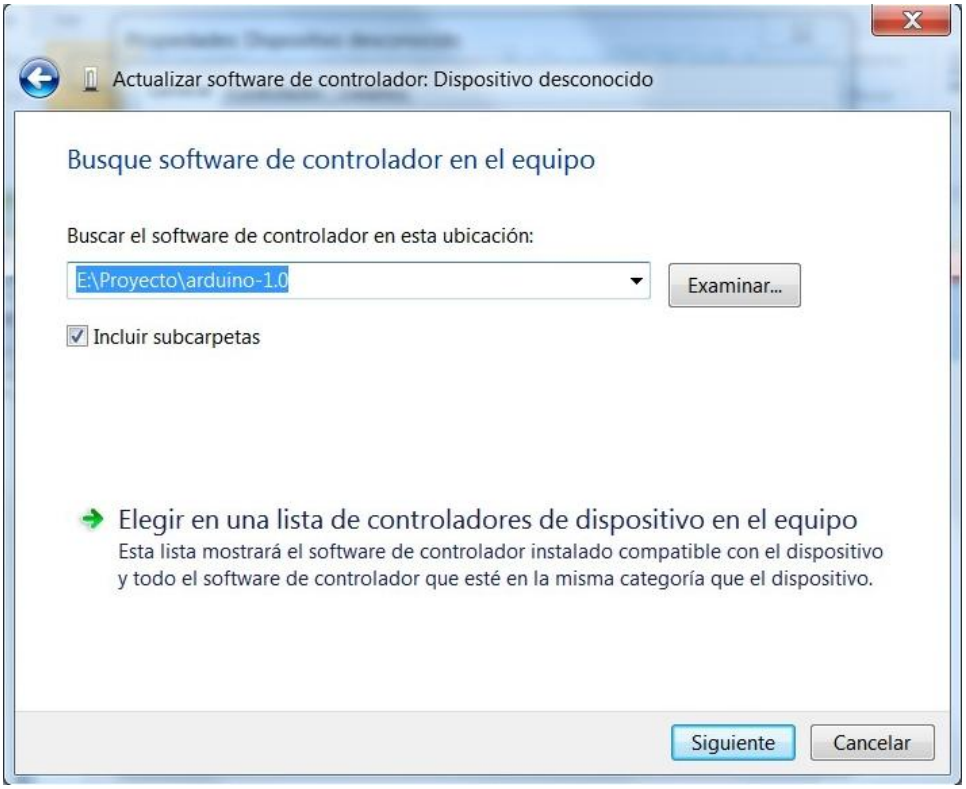


Figura 28 Ubicación Drivers Arduino

- Una vez instalados los drivers podemos observar que ya no aparece como dispositivo desconocido, sino que se le ha asignado un puerto COM, como se puede observar en la Figura 29.

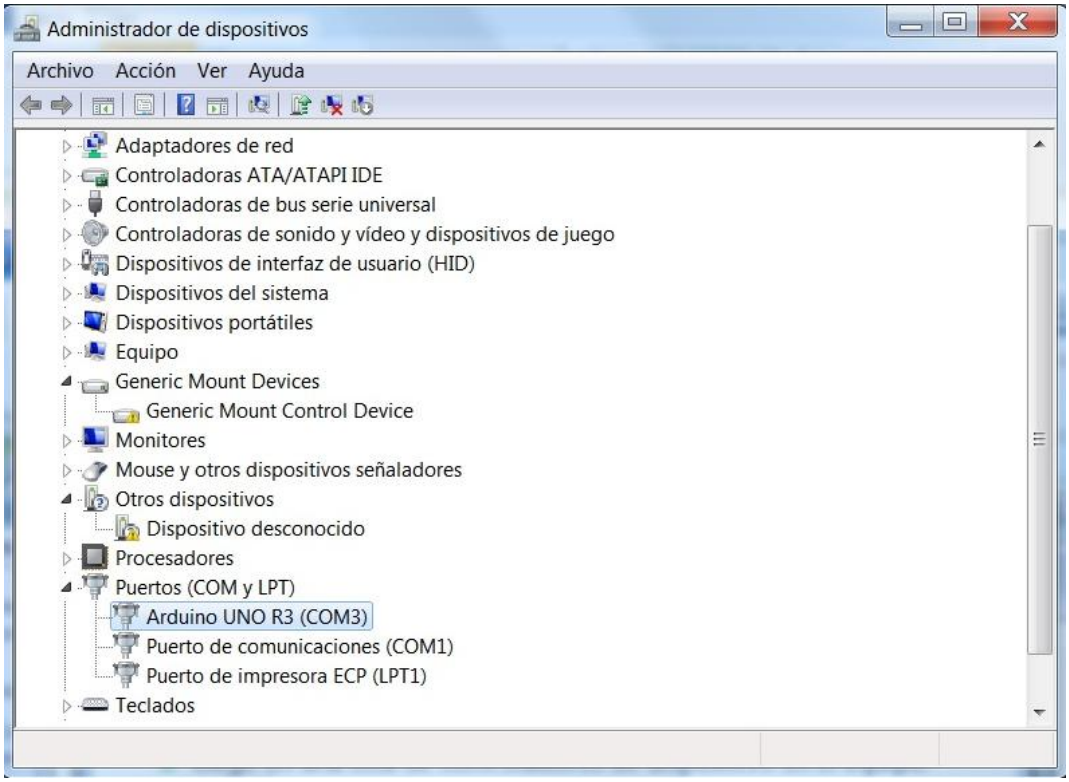


Figura 29 Administrador de dispositivos Arduino UNO R3 Puerto COM

3. *Entorno de programación*

El entorno de programación Arduino 1.0 (ver Figura 30), es un entorno de programación en C.

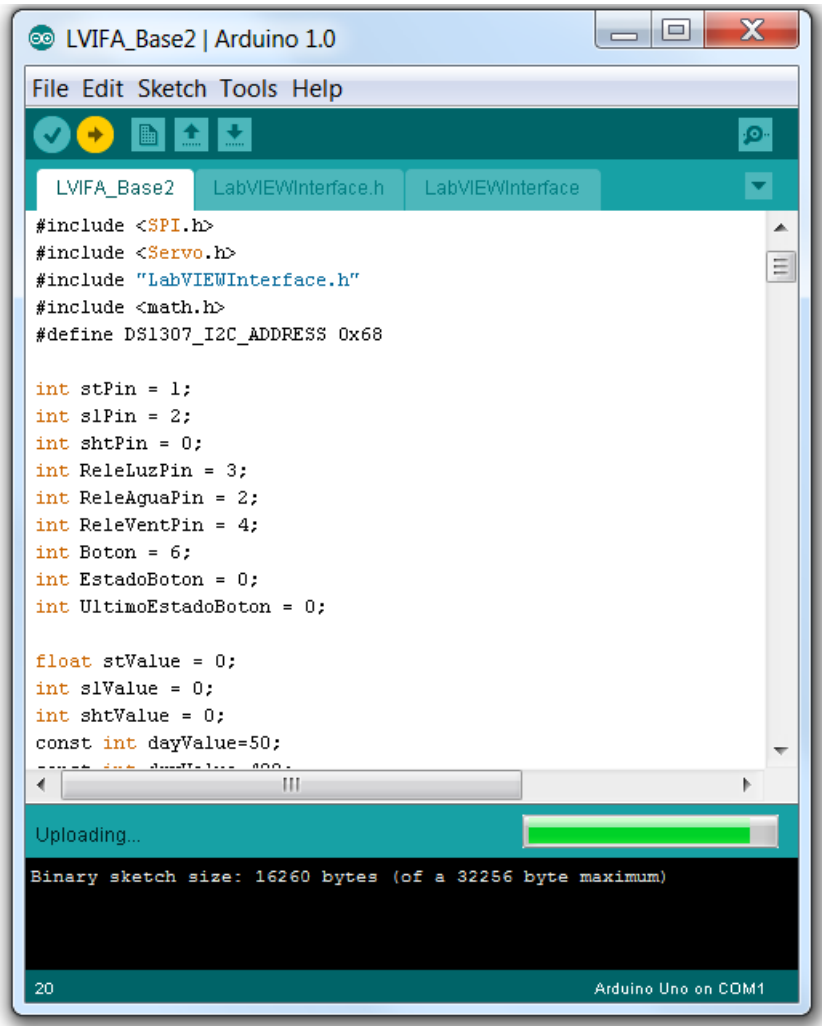
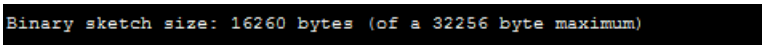


Figura 30 Arduino 1.0

A continuación se va a exponer cada parte del entorno de programación:

- Verify: Compila el programa.
- Upload: Carga el programa en la placa Arduino
- New: Crea un nuevo sketch.
- Open: Acceso rápido a abrir sketch y ejemplos.
- Save: Guarda el sketch actual.
- Serial Monitor: Acceso directo al monitor serie.
- LVIFA_Base2: Nos indica en que pestaña estamos del sketch.
- : Es un menú desplegable para la administración de pestaña.
- Uploading... : Nos indica lo que se está haciendo en estos momentos.
- : Barra de progreso.

- 

Cuadro donde se nos indica los errores y el estado de la memoria.
- 

: Nos indica tanto la placa como el puerto que se está usando actualmente.

4. Lenguaje de programación

Como se comentó anteriormente, el lenguaje de programación es Arduino, está basado en C con funcionalidades de C++.

A continuación vamos a mostrar una tabla con los comandos de estructura utilizados en Arduino:

- Estructura
    - setup() (inicialización)
    - loop() (bucle)
- Estructuras de control
    - if (comparador si-entonces)
    - if...else (comparador si...sino)
    - for (bucle con contador)
    - switch case (comparador múltiple)
    - while (bucle por comparación booleana)
    - do... while (bucle por comparación booleana)
    - break (salida de bloque de código)
    - continue (continuación en bloque de código)
    - return (devuelve valor a programa)
- Sintaxis
    - ; (punto y coma)

- {} (llaves)
- // (comentarios en una línea)
- /* */ (comentarios en múltiples líneas)

- Operadores Aritméticos

  - = (asignación)
  - + (suma)
  - - (resta)
  - (multiplicación)
  - / (división)
  - % (resto)

- Operadores Comparativos

  - == (igual a)
  - != (distinto de)
  - < (menor que)
  - > (mayor que)
  - <= (menor o igual que)
  - >= (mayor o igual que)

- Operadores Booleanos

  - && (y)
  - || (o)
  - ! (negación)

- Operadores de Composición

  - ++ (incrementa)
  - -- (decrementa)
  - += (composición suma)

    - -= (composición resta)
    - *= (composición multiplicación)
    - /= (composición división)

A continuación vamos a exponer los comandos relacionados con las variables.

- Constantes

    - HIGH | LOW (Alto y bajo en entradas y salidas digitales)
    - INPUT | OUTPUT (Entrada y salida en pin digital)
    - true | false (Verdadero falso)

- Tipos de Datos

    - boolean (booleano)
    - char (carácter)
    - byte
    - int (entero)
    - unsigned int (entero sin signo)
    - long (entero 32b)
    - unsigned long (entero 32b sin signo)
    - float (en coma flotante)
    - double (en coma flotante de 32b)
    - string (cadena de caracteres)
    - array (cadena)
    - void (vacío)

Y por último se exponen las funciones más utilizadas en Arduino:

- E/S Digitales

    - pinMode() (definimos el modo de ese pin como entrada o salida)
    - digitalWrite() (Escribe en un pin digital el estado alto o bajo)

    - digitalRead() (Le el estado de un pin digital)

- E/S Analógicas

    - analogRead() (Escribe en un pin analógico)
    - analogWrite() (Le el valor de un pin analógico)

- Tiempo

    - millis()
    - micros()
    - delay()
    - delayMicroseconds()

- Matemáticas

    - min() (mínimo)
    - max() (máximo)
    - abs() (valor absoluto)
    - constrain() (limita)
    - map() (cambia valor de rango)
    - pow() (eleva a un número)
    - sq() (eleva al cuadrado)
    - sqrt() (raíz cuadrada)

- Trigonometría

    - sin() (seno)
    - cos() (coseno)
    - tan() (tangente)

- Números Aleatorios

- randomSeed()
- random()

- Comunicación

  - Serial

5. *Programación del sistema*

En este punto se expone la programación realizada en el proyecto. La programación cuenta con diferentes pasos que se muestran a continuación:

- Se preparara el código para su conexión con 'LabVIEW'. En LabVIEW vamos a utilizar el tookit LIFA (LabVIEW Interface for Arduino), el cual incluye el código a utilizar para la conexión de LabVIEW.
- Diseñar el código para cada componente.
- Unir todo el código en un solo programa.

1. Código LVIFA para LabVIEW

En este punto se expone el código LVIFA_Base para arduino. Este código es necesario para la conexión de Arduino y LabVIEW, por medio del toolkit LIFA.

Este código se encuentra en la carpeta de instalación de LIFA en LabVIEW y es un código de libre distribución y modificación, creado originalmente por Sam Kristoff en noviembre de 2010.

El código consta de 3 pestañas.

- LVIFA_Base : Esta pestaña proporciona un Sketch básico para interactuar con LabVIEW.
- LabVIEWInterface.h: Esta pestaña es una librería con las funciones necesarias para interactuar con LabVIEW.

- LabVIEWInterface: Esta pestaña desarrolla las funciones para interactuar con LabVIEW.

La pestaña **LVIFA_Base** (Ver Figura 31), es una pestaña preparada para introducir el programa que interactúe con LabVIEW, como en este proyecto sucede.

```
/*****************************************************************************
 ** Includes.
 *****************************************************************************/
// Standard includes.  These should always be included.
#include <Wire.h>
#include <SPI.h>
#include <Servo.h>
#include "LabVIEWInterface.h"
/*****************************************************************************
 **  setup()
 **  Initialize the Arduino and setup serial communication.
 **  Input:  None
 **  Output: None
 *****************************************************************************/
void setup()
{
  // Initialize Serial Port With The Default Baud Rate
  syncLV();

  // Place your custom setup code here

}
/*****************************************************************************
 **  loop()
 **  The main loop.  This loop runs continuously on the Arduino.  It
 **  receives and processes serial commands from LabVIEW.
 **  Input:  None
 **  Output: None
 *****************************************************************************/
void loop()
{
  // Check for commands from LabVIEW and process them.

  checkForCommand();
  // Place your custom loop code here (this may slow down communication with LabVIEW)


  if(acqMode==1)
  {
    sampleContinously();
  }

}
```

Figura 31 Pestaña LVIFA_Base Arduino

Se divide en 3 bloques. Un primer bloque donde se definen los includes, las variables globales y en caso de hacer falta, algún programa corto para realizar alguna operación.

Un segundo bloque en el que encontramos el setup(), este bloque se utiliza para inicializar el programa, definir los pin digitales y algún otro pin especial como el I2C, utilizado para el reloj digital.

Y por último el void loop(), es propiamente donde se colocara el programa, debido a que es un bucle, y será el programa que se repita constantemente.

La segunda pestaña es **LabVIEWInterface.h,** es una librería donde se definen las variables globales y a utilizar en esta pestaña (Ver Figura 32) y las funciones utilizadas en la pestaña LabVIEWInterface (ver Figura 33).

```
**  Define directives providing meaningful names for constant values.
*****************************************************************************/

#define FIRMWARE_MAJOR 02
#define FIRMWARE_MINOR 00
#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
#define DEFAULTBAUDRATE 9600    // Defines The Default Serial Baud Rate (This must match the baud rate specifid in LabVIEW)
#else
#define DEFAULTBAUDRATE 115200
#endif
#define MODE_DEFAULT 0            // Defines Arduino Modes (Currently Not Used)
#define COMMANDLENGTH 15          // Defines The Number Of Bytes In A Single LabVIEW Command
//#define STEPPER_SUPPORT 1       // Defines Whether The Stepper Library Is Included-Comment This Line To Exclude Stepper Support


// Declare Variables
unsigned char currentCommand[COMMANDLENGTH];    // The Current Command For The Arduino To Process
//Globals for continuous aquisition
unsigned char acqMode;
unsigned char contAcqPin;
float contAcqSpeed;
float acquisitionPeriod;
float iterationsFlt;
int iterations;
float delayTime;
```

Figura 32 LabVIEWInterface.h: Definición Variables

```
/*****************************************************************************
**  syncLV
**
**  Synchronizes with LabVIEW and sends info about the board and firmware (Unimplemented)
**
**  Input:  None
**  Output: None
*****************************************************************************/
void syncLV();

/*****************************************************************************
**  setMode
**
**  Sets the mode of the Arduino (Reserved For Future Use)
**
**  Input:  Int - Mode
**  Output: None
*****************************************************************************/
void setMode(int mode);

/*****************************************************************************
**  checkForCommand
**
**  Checks for new commands from LabVIEW and processes them if any exists.
**
**  Input:  None
**  Output: 1 - Command received and processed
**          0 - No new command
*****************************************************************************/
int checkForCommand(void);
```

Figura 33 LabVIEWInterface.h: Ejemplo definición de funciones

Esta pestaña no se va modificar en este proyecto. Se modificaría para añadir funciones a utilizar en la tercera pestaña (LabVIEWInterface).

Y por último la pestaña **LabVIEWInterface**. En esta pestaña se van a desarrollar las funciones definidas en la librería LabVIEWInterface.h. La pestaña se divide principalmente en 2 partes. Una primera en la cual se definen las librerías y variables a utilizar en la pestaña (Ver Figura 34)

```
#include <Wire.h>
#include <SPI.h>
#include <LiquidCrystal.h>
#include <math.h>
// Variables

unsigned int retVal;
int sevenSegmentPins[8];
int currentMode;
unsigned int freq;
unsigned long duration;
int i2cReadTimeouts = 0;
char spiBytesToSend = 0;
char spiBytesSent = 0;
char spiCSPin = 0;
char spiWordSize = 0;
Servo *servos;
byte customChar[8];
LiquidCrystal lcd(0,0,0,0,0,0,0);
```

Figura 34 LabVIEWInterface: Definición de librerías y variables

Y una segunda, en la cual se programan las funciones definidas en la librería LabVIEWInterface.h (ver Figura 35).

```
// Sets the mode of the Arduino (Reserved For Future Use)
void setMode(int mode)
{
  currentMode = mode;
}

// Checks for new commands from LabVIEW and processes them if any exists.
int checkForCommand(void)
{

  int bufferBytes = Serial.available();

  if(bufferBytes >= COMMANDLENGTH)
  {
    // New Command Ready, Process It
    // Build Command From Serial Buffer
    for(int i=0; i<COMMANDLENGTH; i++)
    {
      currentCommand[i] = Serial.read();
    }
    processCommand(currentCommand);
    return 1;
  }
  else
  {
    return 0;
  }
}
```

Figura 35 LabVIEWInterface: Ejemplo programación funciones

Para modificar esta pestaña hay dos opciones. Definir una nueva función en la librería LabVIEWInterface.h y programarla es la pestaña LabVIEWInterface, o añadir un case nuevo dentro de la función processCommand (Ver Figura 36).

```
// Processes a given command
void processCommand(unsigned char command[])
{
  // Determine Command
  if(command[0] == 0xFF && checksum_Test(command) == 0)
  {
    switch(command[1])
    {
    /*****************************************************************************
    ** LIFA Maintenance Commands
    *****************************************************************************/
    case 0x00:     // Sync Packet
      Serial.print("sync");
      Serial.flush();
      break;
    case 0x01:    // Flush Serial Buffer
      Serial.flush();
      break;

    /*****************************************************************************
    ** Low Level - Digital I/O Commands
    *****************************************************************************/
    case 0x02:    // Set Pin As Input Or Output
      pinMode(command[2], command[3]);
      Serial.write('0');
      break;
    case 0x03:    // Write Digital Pin
      digitalWrite(command[2], command[3]);
       Serial.write('0');
      break;
    case 0x04:    // Write Digital Port 0
      writeDigitalPort(command);
       Serial.write('0');
      break;
```

Figura 36 LabVIEWInterface: Función proccesCommand ejemplo case.

2. Programación por componentes.

En este punto se muestra el código a implementar para cada componente. Posterior mente se mostrara el código completo.

- Módulo sensor de luz conectar y listo (Ver Figura 37):

```
#include <math.h>

int slPin = 2;
int ReleLuzPin = 3;
float slValue = 0;
const int dayValue=50;

void setup()
{
    Serial.begin(115200);
    pinMode(ReleLuzPin,OUTPUT);

}

void loop()
{

slValue = analogRead(slPin);
slValue = (slValue)/10;

if(slValue<dayValue)
{
  digitalWrite(ReleLuzPin,HIGH);
}
else
{
  digitalWrite(ReleLuzPin,LOW);
}

Serial.print("Luminosidad (0-100%) = ");
Serial.println(slValue);

delay(1000);

}
```

Figura 37 Código módulo de luz conectar y listo

El código comienza con la definición de la librería necesaria (math.h). A continuación definimos las variables necesarias:

- slPin: El pin analógico donde conectamos el sensor.
- ReleLuzPin: El pin digital donde conectamos el relé asociado a la luz.
- slValue: El valor que manejaremos en las operaciones.
- dayValue: El valor de luminosidad límite para activar la luz.

Dentro del void setup() se encuentra:

- Serial.begin(115200): Definimos la salida serie.
- pinMode(ReleLuzPin,OUTPUT): Definimos el pin digital del relé asignado a luz como un pin de salida.

Y por último dentro del void loop() encontramos:

- slValue: analogRead(slPin): Asignamos a la variable slValue el valor del pin analógico.
- slValue= (slValue)/10: Convertimos el valor a escala 0-100%
- if(slValue<dayValue): Si el valor de luminosidad es menor al valor estimado dayValue,
- digitalWrite(ReleLuzPin,HIGH): Escribe en el pin digital del relé asignado a luz el valor alto (se activa el rele)
- else: de lo contrario a if(slValue<dayValue).
- digitalWrite(ReleLuzPin,LOW): Escribe en el pin digital del relé asignado a luz el valor bajo (se desactiva el relé)
- Serial.print("Luminosidad(0-100%)= "); Serial.println(slValue): Nos escribe la salida serie el texto "Luminosidad (0-100%) = xx.xx"
- delay(1000): Para el programa el tiempo de 1000ms.

- Módulo sensor de temperatura conectar y listo (ver Figura 38):

```
#include <math.h>

int stPin = 1;
int ReleVentPin = 4;
int ReleCalPin = 9;
float stValue = 0;
float Resistance = 0;
const int hotValue=26;
const int coldValue=5;

void setup()
{
  Serial.begin(115200);
  pinMode(ReleVentPin,OUTPUT);
  pinMode(ReleCalPin,OUTPUT);
}

void loop()
{
  stValue = analogRead(stPin);
  Resistance=(float)(1023-stValue)*10000/stValue;
  stValue=1/(log(Resistance/10000)/3975+1/298.15)-273.15;

  if(stValue>hotValue)
  {
    digitalWrite(ReleVentPin,HIGH);
  }
  else
  {
    digitalWrite(ReleVentPin,LOW);
  }
  if(stValue<coldValue)
  {
    digitalWrite(ReleCalPin,HIGH);
  }
  else
  {
    digitalWrite(ReleCalPin,LOW);
  }
  Serial.print("Temperatura = ");
  Serial.print(stValue);
  Serial.println("C  ");

  delay(1000);
}
```

Figura 38 Código módulo de temperatura conectar y listo

El código comienza con la definición de la librería necesaria (math.h). A continuación definimos las variables necesarias:

- stPin: El pin analógico donde conectamos el sensor.
- ReleVentPin: El pin digital donde conectamos el relé asociado a los ventiladores.

- ReleCalPin: El pin digital donde conectamos el relé asociado a la calefacción.
- stValue: El valor que manejaremos en las operaciones.
- Resistance: El valor que asignaremos a la resistencia del termistor.
- hotValue: El valor de temperatura superior límite para activar los ventiladores.
- coldValue: El valor de temperatura inferior límite para activar el calefactor.

Dentro del void setup() se encuentra:

- Serial.begin(115200): Definimos la salida serie.
- pinMode(ReleVentPin,OUTPUT): Definimos el pin digital del relé asignado a los ventiladores como un pin de salida.
- pinMode(ReleCalPin,OUTPUT): Definimos el pin digital del relé asignado a la calefacción como un pin de salida.

Y por último dentro del void loop() encontramos:

- stValue: analogRead(slPin): Asignamos a la variable stValue el valor del pin analógico.
- Resistance=(float)(1023-stValue)*10000/stValue: Calculamos el valor de la resistencia del termistor.
- stValue=1/(log(Resistance/10000)/3975+1/298.15)-273.15: Calculamos la temperatura en ºC.
- if(stValue>hotValue): Si el valor de la temperatura es mayor al valor estimado hotValue,
- digitalWrite(ReleVentPin,HIGH): Escribe en el pin digital del relé asignado a los ventiladores el valor alto (se activa el relé).
- else: de lo contrario a if(stValue>hotValue).
- digitalWrite(ReleVentPin,LOW): Escribe en el pin digital del relé asignado a los ventiladores el valor bajo (se desactiva el relé)

- if(stValue<coldValue): Si el valor de la temperatura es menor al valor estimado coldValue,
- digitalWrite(ReleCalPin,HIGH): Escribe en el pin digital del relé asignado a la calefacción el valor alto (se activa el relé).
- else: de lo contrario a if(stValue<coldValue).
- digitalWrite(ReleCalPin,LOW): Escribe en el pin digital del relé asignado a la calefacción el valor bajo (se desactiva el relé)
- Serial.print("Temperatura =);Serial.print(stValue);Serial.println("C") Nos escribe la salida serie el texto "Temperatura = xx.xxC"
- delay(1000): Para el programa el tiempo de 1000ms.

- Módulo sensor Humedad del suelo conectar y listo (ver Figura 39):

```
#include <math.h>

int shtPin = 0;
int ReleAguaPin = 2;
float shtValue = 0;
const int dryValue=40;

void setup()
{
  Serial.begin(115200);
  pinMode(ReleAguaPin,OUTPUT);
}

void loop()
{
  shtValue = analogRead(shtPin);
  shtValue = shtValue/10;

   if(shtValue<dryValue)
  {
    digitalWrite(ReleAguaPin,HIGH);
  }
  else
  {
    digitalWrite(ReleAguaPin,LOW);
  }

  Serial.print("humedad tierra = " );
  Serial.print(shtValue);
  Serial.println("%");

  delay(1000);
}
```

Figura 39 Código módulo humedad suelo conectar y listo

El código comienza con la definición de la librería necesaria (math.h). A continuación definimos las variables necesarias:

- shtPin: El pin analógico donde conectamos el sensor.
- ReleAguaPin: El pin digital donde conectamos el relé asociado al riego.
- shtValue: El valor que manejaremos en las operaciones.
- dryValue: El valor de humedad límite para activar la luz.

Dentro del void setup() se encuentra:

- Serial.begin(115200): Definimos la salida serie.
- pinMode(ReleAguaPin,OUTPUT): Definimos el pin digital del relé asignado al riego como un pin de salida.

Y por último dentro del void loop() encontramos:

- shtValue: analogRead(slPin): Asignamos a la variable shtValue el valor del pin analógico.
- shtValue= (slValue)/10: Convertimos el valor a escala 0-100%
- if(shtValue<dryValue): Si el valor de humedad es menor al valor estimado dryValue,
- digitalWrite(RelAguaPin,HIGH): Escribe en el pin digital del relé asignado al riego el valor alto (se activa el relé)
- else: de lo contrario a if(shtValue<dryValue).
- digitalWrite(ReleAguaPin,LOW): Escribe en el pin digital del relé asignado al riego el valor bajo (se desactiva el relé)
- Serial.print("Humedad del suelo = “); Serial.print(shtValue); Serial.println("%") Nos escribe la salida serie el texto “Humedad del suelo = xx.xx%”
- delay(1000): Para el programa el tiempo de 1000ms.

- Modulo Reloj tiempo real DS1307 conectar y listo.

Este módulo cuenta con el código más complejo de los utilizados en este proyecto. Es un módulo con conexión I^2C (Inter-Integrated Circuit) y su principal característica es que utiliza dos líneas para transmitir información.

Vamos a dividir el código en tres bloques. El primero (ver Figure X), será desde el inicio hasta el void setup().

```
#include <Wire.h>
#include <math.h>
#define DS1307_I2C_ADDRESS 0x68

// Convierte números normales decimales a BCD (binario decimal codificado)
byte decToBcd(byte val)
{
  return ( (val/10*16) + (val%10) );
}

// Convierte BCD (binario decimal codificado) a números normales decimales
byte bcdToDec(byte val)
{
  return ( (val/16*10) + (val%16) );
}

// Establece la fecha y el tiempo del ds1307
void getDateDs1307(byte *second,
          byte *minute,
          byte *hour,
          byte *dayOfWeek,
          byte *dayOfMonth,
          byte *month,
          byte *year)
{
// Resetea el registro puntero
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.write((byte)0);
  Wire.endTransmission();

  Wire.requestFrom(DS1307_I2C_ADDRESS, 7);

  // Alguno de estos necesitan enmascarar porque ciertos bits son bits de control

  *second     = bcdToDec(Wire.read() & 0x7f);
  *minute     = bcdToDec(Wire.read());
  *hour       = bcdToDec(Wire.read() & 0x3f);  // Need to change this if 12 hour am/pm
  *dayOfWeek  = bcdToDec(Wire.read());
  *dayOfMonth = bcdToDec(Wire.read());
  *month      = bcdToDec(Wire.read());
  *year       = bcdToDec(Wire.read());
}
```

Figura 40 Primer bloque código DS1307

El código comienza con la definición de las librerías necesaria (math.h) y (Wire.h). A continuación definimos la dirección del reloj en el puerto I^2C. Posteriormente se incluyen 3 pequeños programas:

- byte decToBcd(byte val){}: Convierte números normales decimales a BCD (codificado binario decimal).
- byte bcdToDec(byte val){}: Convierte de BCD (codificado binario decimal) a números normales decimales.
- Void getDateDs1307(): Aquí se resetea el registro de puntero, y se obtiene la fecha y el tiempo del ds1307.

A continuación se presenta el segundo bloque (ver Figura 41), en el cual vamos a estudiar el void setup().

```
void setup()
{
  byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
  Wire.begin();
  Serial.begin(115200);

  /*
  // si desea poner en hora, active esta parte y luego vuelva a quitarla
  second = 00;
  minute = 15;
  hour = 12;
  dayOfWeek = 2;
  dayOfMonth = 22;
  month = 5;
  year = 12;

  setDateDs1307(second, minute, hour, dayOfWeek, dayOfMonth, month, year);
   */
}
```

Figura 41 Void setup() código DS1307

- byte second, minute, hour, dayOfWeek, dayOfMonth, month, year: Define dentro de void setup() las variables temporales.
- Wire.begin(): Inicia el pin asociado a I^2C.
- Serial.begin(115200): Definimos la salida serie.
- second = XX; minute = XX; hour = XX; dayOfWeek = X; dayOfMonth = XX; month = XX; year = XX; setDateDs1307(second, minute, hour, dayOfWeek, dayOfMonth, month, year): Ponemos en hora el reloj. Para ello se tiene que cargar primeramente esta función sin el comentario, estableciendo la hora, y posteriormente, incluir esta función en comentario y volver a cargar el código.

Por ultimo vamos a estudiar la última parte del código asociado al reloj DS1307 (ver Figura 42, el asociado al void loop().

```
void loop()
{
 byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
  getDateDs1307(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month, &year);

  Serial.print("20");
    if (year < 10) Serial.print("0");
  Serial.print(year, DEC);
  Serial.print("/");
    if (month < 10) Serial.print("0");
  Serial.print(month, DEC);
  Serial.print("/");
    if (dayOfMonth < 10) Serial.print("0");
  Serial.print(dayOfMonth, DEC);
  Serial.print("  ");
    if (hour < 10) Serial.print("0");
  Serial.print(hour, DEC);
  Serial.print(":");
    if (minute < 10) Serial.print("0");
  Serial.print(minute, DEC);
  Serial.print(":");
    if (second < 10) Serial.print("0");
  Serial.print(second, DEC);
  Serial.print("  Dia de la semana:");
//  Serial.println(dayOfWeek, DEC);
// Esto pone nombre del dia
    switch (dayOfWeek)
        {
    case 1:
      Serial.println("  Lunes");       break;
    case 2:
      Serial.println("  Martes");      break;
    case 3:
      Serial.println("  Miercoles");   break;
    case 4:
      Serial.println("  Jueves");      break;
    case 5:
      Serial.println("  Viernes");     break;
    case 6:
      Serial.println("  Sabado");      break;
    case 7:
      Serial.println("  Domingo");     break;
       }
       delay(1000)
}
```

Figura 42 Void loop() código DS1307

- byte second, minute, hour, dayOfWeek, dayOfMonth, month, year: Define dentro de void setup() las variables temporales.
- getDateDs1307(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month, &year): Asigna los valores a las variables temporales en el void loop().

- Con el código siguiente lo que hacemos es imprimir en la salida serie la fecha, hora y día de la semana de la siguiente manera:
  "2012 / 05/ 20 00:00:01 Día de la semana: Domingo"

3. Programa completo

En este punto se va exponer el programa completo implementado en arduino. Se basa en la adaptación del código de cada componente al código LVIFA_Base. Como se explicó en puntos anteriores. En este proyecto vamos a modificar solo la pestaña LVIFA_Base, dejando con el código original las pestañas LabVIEWInterface.h y LabVIEWInterface.

Además de los componentes anteriormente explicados, es necesaria la incorporación en el código de un interruptor para seleccionar el modo automático (sin conexión a pc) o el modo manual (con conexión a pc) que se verá en próximos puntos.

También será necesario adaptar el código para apagar los relés cada vez que se cambie el interruptor de posición, para evitar que se queden encendidos tras el cambio de modo.

En primer lugar se expone la definición de las librerías necesarias para este programa (ver Figura 43) así como la definición de las variables globales necesarias en esta pestaña y la definición de la dirección del reloj en el puerto I^2C (ver Figura 44).

```
#include <Wire.h>
#include <SPI.h>
#include <Servo.h>
#include "LabVIEWInterface.h"
#include <math.h>
```

Figura 43 Definición de librerías LVIFA_Base

```
#define DS1307_I2C_ADDRESS 0x68
int stPin = 1;
int slPin = 2;
int shtPin = 0;
int ReleLuzPin = 3;
int ReleAguaPin = 2;
int ReleVentPin = 4;
int ReleCalPin = 9;
int Boton = 6;
int EstadoBoton = 0;
int UltimoEstadoBoton = 0;

float Time, minTime, maxTime;
float stValue = 0;
float slValue = 0;
float shtValue = 0;
float Resistance;
const int dayValue=50;
const int dryValue=40;
const int hotValue=26;
const int coldValue=5;
const int minHourValue=9;
const int maxHourValue=22;
const int minMinuteValue=30;
const int maxMinuteValue=30;
```

Figura 44 Definición de variables LVIFA_Base

Como se puede observar (ver Figura 44), además de las variables anteriormente vistas, se encuentra la definición de la variable Boton, EstadoBoton y UltimoEstadoBoton que posteriormente veremos su utilización en la definición del interruptor y el programa necesario para el apagado de los relés en el cambio de posición del interruptor.

También se han añadido las variables minHourValue, maxHourValue, minMinuteValue y maxMinuteValue. Estas variables definen la hora y los minutos mínimos y máximos para el encendido de luces.

Por medio de las funciones const int, establecemos los valores mínimos y máximos para el encendido de los relés asociados a riego, luz, ventilación y calefacción. Modificando esos valores se modificaran los valores límites para el encendido y apagado automático de los relés.

A continuación se colocan las funciones utilizadas en el reloj DS1307 (ver Figura 45) para la conversión entre BCD (binario decimal codificado) y los números normales decimales, así como la función que resetea el registro de puntero y obtiene la fecha y el tiempo.

```
byte decToBcd(byte val)
{
  return ( (val/10*16) + (val%10) );
}

byte bcdToDec(byte val)
{
  return ( (val/16*10) + (val%16) );
}

void getDateDs1307(byte *second,
          byte *minute,
          byte *hour,
          byte *dayOfWeek,
          byte *dayOfMonth,
          byte *month,
          byte *year)
{
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.write((byte)0);
  Wire.endTransmission();

  Wire.requestFrom(DS1307_I2C_ADDRESS, 7);

  *second     = bcdToDec(Wire.read() & 0x7f);
  *minute     = bcdToDec(Wire.read());
  *hour       = bcdToDec(Wire.read() & 0x3f);
  *dayOfWeek  = bcdToDec(Wire.read());
  *dayOfMonth = bcdToDec(Wire.read());
  *month      = bcdToDec(Wire.read());
  *year       = bcdToDec(Wire.read());
}
```

Figura 45 Funciones DS1307 en LVIFA_Base

Después de las funciones para DS1307 se ubica el void setup() (ver Figura 46). Se puede observar que se ha sustituido el comando Serial.begin(115200) por la llamada a la función syncLV() que se encuentra en la librería LabVIEWInterface.h y se desarrolla en pestaña LabVIEWInterface (ver Figura X).

```
void setup()
{
  byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
  Wire.begin();
  syncLV();

  pinMode(ReleLuzPin,OUTPUT);
  pinMode(ReleAguaPin,OUTPUT);
  pinMode(ReleVentPin,OUTPUT);
  pinMode(ReleCalPin,OUTPUT);
  pinMode(Boton,INPUT);

  /*
  // si desea poner en hora, active esta parte y luego vuelva a quitarla
  second = 00;
  minute = 15;
  hour = 12;
  dayOfWeek = 2;
  dayOfMonth = 22;
  month = 5;
  year = 12;

  setDateDs1307(second, minute, hour, dayOfWeek, dayOfMonth, month, year);
   */
}
```

Figura 46 Función void setup() en LVIFA_Base

```
// Synchronizes with LabVIEW and sends info about the board and firmware
void syncLV()
{
  Serial.begin(DEFAULTBAUDRATE);
  i2cReadTimeouts = 0;
  spiBytesSent = 0;
  spiBytesToSend = 0;
  Serial.flush();
}
```

Figura 47 Función syncLV() para void setup() en LVIFA_Base

Por último se coloca la función void loop(). Esta parte del código es la que más varía con respecto a la programación individual de cada componente. Hay que adaptar, primero el código de cada componente al código LVIFA_Base. Además aquí se programa el interruptor para la selección entre funcionamiento automático (sin conexión a pc) y manual (conexión a pc). Y también se implementa el código necesario para el apagado de los relés tras cada cambio de posición del interruptor para la selección de funcionamiento.

```
void loop()
{

  checkForCommand();

  if(digitalRead(Boton)==HIGH)
  {
  EstadoBoton = 0;
  if (EstadoBoton != UltimoEstadoBoton)
  {
    digitalWrite(ReleAguaPin,LOW);
    digitalWrite(ReleLuzPin,LOW);
    digitalWrite(ReleVentPin,LOW);
    digitalWrite(ReleCalPin,LOW);
  }
```

Figura 48 void loop() para LVIFA_Base. Primera parte

La función void loop() comienza con el comando checkForCommand() perteneciente a la librería LabVIEWInterface.h y desarrollado en la pestaña LabVIEWInterface (ver Figura 49). A continuación se encuentra la selección de función por medio del interruptor, para el funcionamiento automático (sin conexión pc) se utiliza la posición HIGH del interruptor y se le asocia el valor 0 a la variable EstadoBoton (ver Figura 48).

```
// Checks for new commands from LabVIEW and processes them if any exists.
int checkForCommand(void)
{

  int bufferBytes = Serial.available();

  if(bufferBytes >= COMMANDLENGTH)
  {
    // New Command Ready, Process It
    // Build Command From Serial Buffer
    for(int i=0; i<COMMANDLENGTH; i++)
    {
      currentCommand[i] = Serial.read();
    }
    processCommand(currentCommand);
    return 1;
  }
  else
  {
    return 0;
  }
}
```

Figura 49 Función checkForCommand() para void loop() en LVIFA_Base

Y por último dentro de esta primera parte del bloque void loop() se encuentra el código necesario para el apagado de los relés para el cambio de posición en el

interruptor. El código se basa en la comparación de las variables EstadoBoton y UltimoEstadoBoton. En caso de ser diferentes se apagan automáticamente los relés. Como se verá posteriormente (ver Figura 50) el valor de UltimoEstadoBoton se asigna al final del código correspondiente al valor HIGH en la variable Boton.

```
byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;

getDateDs1307(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month, &year);

Serial.print("20");
  if (year < 10) Serial.print("0");
Serial.print(year, DEC);
Serial.print("/");
  if (month < 10) Serial.print("0");
Serial.print(month, DEC);
Serial.print("/");
  if (dayOfMonth < 10) Serial.print("0");
Serial.print(dayOfMonth, DEC);
Serial.print("  ");
  if (hour < 10) Serial.print("0");
Serial.print(hour, DEC);
Serial.print(":");
  if (minute < 10) Serial.print("0");
Serial.print(minute, DEC);
Serial.print(":");
  if (second < 10) Serial.print("0");
Serial.print(second, DEC);
Serial.print("  Dia de la semana:");

  switch (dayOfWeek)
      {
  case 1:
    Serial.println("  Lunes");       break;
  case 2:
    Serial.println("  Martes");      break;
  case 3:
    Serial.println("  Miercoles");   break;
  case 4:
    Serial.println("  Jueves");      break;
  case 5:
    Serial.println("  Viernes");     break;
  case 6:
    Serial.println("  Sabado");      break;
  case 7:
    Serial.println("  Domingo");     break;
 }
```

Figura 50 void loop() para LVIFA_Base. Segunda parte

La segunda parte del void loop() es el código void loop() del módulo DS1307 (ver Figura 51) sin ninguna variación con respecto a lo visto anteriormente.

La tercera parte del void loop() la conforma el código referente a los sensores, activación de relés, y la impresión en la salida serie de los datos obtenidos por los mismo.

```
slValue = analogRead(slPin);
shtValue = analogRead(shtPin);
stValue = analogRead(stPin);

Resistance=(float)(1023-stValue)*10000/stValue;
stValue=1/(log(Resistance/10000)/3975+1/298.15)-273.15;

slValue = slValue/10;
shtValue = shtValue/10;

minTime=minHourValue+minMinuteValue/60;
maxTime=maxHourValue+maxMinuteValue/60;
Time=hour+minute/60;

 if(slValue<dayValue  && Time>=minTime && Time<=maxTime)
{
  digitalWrite(ReleLuzPin,HIGH);
}
else
{
digitalWrite(ReleLuzPin,LOW);
}

 if(shtValue<dryValue)
{
  digitalWrite(ReleAguaPin,HIGH);
}
else
{
  digitalWrite(ReleAguaPin,LOW);
}

if(stValue>hotValue)
{
  digitalWrite(ReleVentPin,HIGH);
}
else
{
  digitalWrite(ReleVentPin,LOW);
}

f(stValue<coldValue)
{
  digitalWrite(ReleCalPin,HIGH);
}
else
{
  digitalWrite(ReleCalPin,LOW);
}

Serial.print("Temperature = ");
Serial.print(stValue);
Serial.print("C  ");
Serial.print("humedad tierra = " );
Serial.print(shtValue);
Serial.print("%");
Serial.print("  Luminosidad (0-100%) = " );
Serial.println(slValue);

UltimoEstadoBoton = EstadoBoton;
delay(1000);
}
```

Figura 51 void loop() para LVIFA_Base. Tercera parte

No se han realizado cambios con respecto a lo visto para cada componente anteriormente, solo se han unido los códigos relativos a cada componente en un único código.

Para finalizar el void loop() y dar por terminada la pestaña LVIFA_Base se expone la función para el estado de la variable boton LOW, correspondiente al funcionamiento manual (ver Figura 52). Esta última parte comienza con la asignación a la variable EstadoBoton del valor 1, y continua con la función de apagado automático de los relés para el cambio de posición del interruptor.

```
  if(digitalRead(Boton)==LOW)
  {
  EstadoBoton = 1;

  if (EstadoBoton != UltimoEstadoBoton)
  {
    digitalWrite(ReleAguaPin,LOW);
    digitalWrite(ReleLuzPin,LOW);
    digitalWrite(ReleVentPin,LOW);
    digitalWrite(ReleCalPin,LOW);
  }
    if(acqMode==1)
  {
    sampleContinously();
  }
   UltimoEstadoBoton = EstadoBoton;
  }
}
```

Figura 52 void loop() para LVIFA_Base. Cuarta parte

Tras la función de apagado automático de relés, se ubica la última parta del código LVIFA_Base original, en la cual se llama a la función sampleContinously() (ver Figura 53) correspondiente a la librería LabVIEWInterface.h y desarrollado en la pestaña LabVIEWInterface, la cual se encarga del muestreo continuo y comunicación con arduino.

```
void sampleContinously()
{

  for(int i=0; i<iterations; i++)
  {
     retVal = analogRead(contAcqPin);
     if(contAcqSpeed>1000)
     {
       Serial.write( (retVal >> 2));
       delayMicroseconds(delayTime*1000000);
     }
     else
     {
        Serial.write( (retVal & 0xFF) );
        Serial.write( (retVal >> 8));
        delay(delayTime*1000);
     }
  }
}
```

Figura 53 Función sampleContinously() para void loop() en LVIFA_Base

Por último, se muestra una Figura de la salida que se obtiene en el serial monitor en caso de conectar Arduino al ordenador.

```
2012/06/08  11:35:33  Dia de la semana:  Viernes
Temperature = 22.27C  humedad tierra = 52.10%  Luminosidad (0-100%) = 74.20
2012/06/08  11:35:34  Dia de la semana:  Viernes
Temperature = 22.27C  humedad tierra = 52.10%  Luminosidad (0-100%) = 74.30
2012/06/08  11:35:35  Dia de la semana:  Viernes
Temperature = 22.27C  humedad tierra = 52.10%  Luminosidad (0-100%) = 74.10
2012/06/08  11:35:36  Dia de la semana:  Viernes
Temperature = 22.27C  humedad tierra = 52.10%  Luminosidad (0-100%) = 74.30
2012/06/08  11:35:37  Dia de la semana:  Viernes
Temperature = 22.27C  humedad tierra = 52.10%  Luminosidad (0-100%) = 74.20
2012/06/08  11:35:38  Dia de la semana:  Viernes
Temperature = 22.27C  humedad tierra = 52.10%  Luminosidad (0-100%) = 74.30
2012/06/08  11:35:39  Dia de la semana:  Viernes
Temperature = 22.27C  humedad tierra = 52.10%  Luminosidad (0-100%) = 74.20
2012/06/08  11:35:40  Dia de la semana:  Viernes
Temperature = 22.27C  humedad tierra = 52.10%  Luminosidad (0-100%) = 74.10
2012/06/08  11:35:41  Dia de la semana:  Viernes
Temperature = 22.27C  humedad tierra = 52.10%  Luminosidad (0-100%) = 74.20
2012/06/08  11:35:42  Dia de la semana:  Viernes
Temperature = 22.27C  humedad tierra = 52.10%  Luminosidad (0-100%) = 74.20
2012/06/08  11:35:43  Dia de la semana:  Viernes
Temperature = 22.27C  humedad tierra = 52.10%  Luminosidad (0-100%) = 74.30
2012/06/08  11:35:44  Dia de la semana:  Viernes
Temperature = 22.27C  humedad tierra = 52.10%  Luminosidad (0-100%) = 74.30
2012/06/08  11:35:45  Dia de la semana:  Viernes
Temperature = 22.27C  humedad tierra = 52.10%  Luminosidad (0-100%) = 74.20
```

Figura 54 Serial Monitor Arduino

Visto esto, se da por terminado la programación de arduino, que en caso de ser necesario se puede adaptar a nuevos componentes a la necesidad de incluir nuevos sensores de humedad, luminosidad y temperatura para grandes espacios. Adaptando el código individual del componente al código global, de la manera que se ha explicado a lo largo de este apartado.

3. Programación LabVIEW

1. Introducción

LabVIEW es una herramienta de programación gráfica, que permite la construcción de sistemas de adquisición de datos, instrumentación, control y test. A través de este lenguaje podemos crear rápidamente una interfaz de usuario para interactuar con el sistema.

El entorno de programación LabVIEW se estructura de la forma siguiente:

- La interactividad con el usuario se realiza a través de un VI (Virtual Instrument), que simula el panel del instrumento físico. Este VI se diseña en el Panel Frontal. Este panel frontal puede contener botones, interruptores, pulsadores, gráficas y otros controles e indicadores. Los datos se introducen utilizando el ratón y el teclado, y los resultados se muestran en la pantalla del ordenador.

  El VI recibe las instrucciones programadas del Diagrama de Bloques que construimos utilizando el lenguaje de programación G (Graphic). El diagrama de bloques es el código fuente de nuestro programa o VI.

- Los VIs son jerárquicos y modulares. Pueden utilizarse como programas o como subprogramas de otros programas. Cuando un VI se usa dentro de otro VI, se denominan subVI. El icono y los conectores de un VI funcionan como una lista de parámetros gráficos de forma que otros VIs puedan pasar datos a otro determinado subVI.

2. Comunicación Arduino-LabVIEW: LIFA toolkit

La comunicación Arduino-LabVIEW se va realizar mediante la conexión USB del ordenador y de la tarjeta arduino. El cable utilizado es cable USB 2.0 con terminales A y B.

La librería utilizada en LabVIEW para la comunicación con la placa arduino va ser la toolkit LIFA (LabVIEW Interfaz For Arduino) como se comentó en apartados

anteriores. Esta toolkit nos proporciona una serie de subVI especialmente diseñadas para la placa arduino, de forma que solo tengamos que indicarle una serie de variables para su conexión.

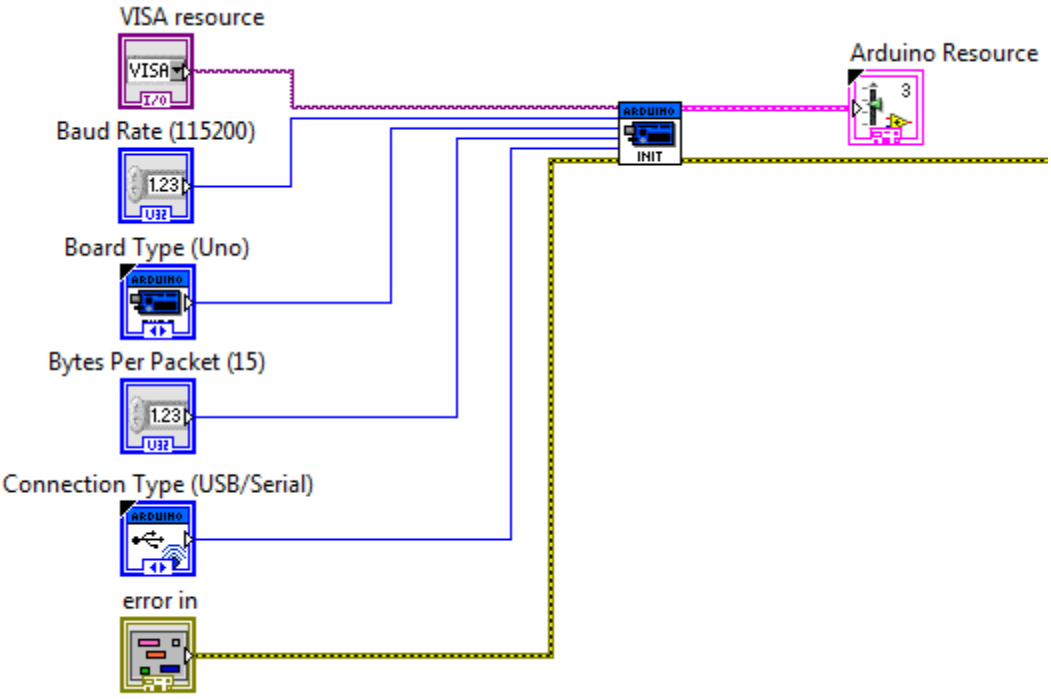


Figura 55 Entradas y salidas SubVI Init.vi

Para iniciar la comunicación con la placa arduino, se ha de colocar la subVI Init.vi. Esta subVI cuenta con 6 variables de entrada y 2 salidas (ver Figura 55) con las que definimos:

- VISA resource: Indicamos el puerto COM al cual tenemos conectada la placa. En nuestro caso es el COM4 en el cual hemos conectado la placa arduino. Este campo es obligatorio de introducir al no contar con ningún valor predefinido.
- Baud Rate (115200): La tasa de baudios también conocida como baudaje, es el número de unidades de señal por segundo. Viene establecida de forma predeterminada en 115200, de forma que en caso de no introducir ningún otro valor utilizaría el predeterminado. En nuestro proyecto se usa el 115200.

- Board Tipe (Uno): Indicamos el tipo de placa que estamos conectando. Viene predeterminado para la placa Uno, de forma que en caso de no introducir ninguna otra placa, actuara como si la placa conectada sea Arduino UNO. Nosotros utilizamos la placa Arduino UNO.
- Connection Type (USB/Serial): Indicamos el tipo de conexión utilizado en la computadora para la conexión arduino. Viene predeterminado para conexión USB/Serial, pero en caso de utilizar otra entrada deberíamos comunicárselo. En nuestro caso utilizaremos el USB.
- Error in: Es la entrada de errores.
- Arduino Resource: Es la salida que nos muestra la información obtenida en el arduino. Se utilizara en otras funciones como entrada.
- Error out: La salida de errores.

A su vez, también es necesario terminar la comunicación con arduino, y esto se realiza por medio del subVI Close.vi. Cuenta con dos entradas y una salida (ver Figura 56). Las entradas y salidas son:

- Arduino Resource: Salida de información obtenida de Init.vi y que utilizaremos en diversas funciones.
- Error in: Es la entrada de errores.
- Error out: La salida de errores.

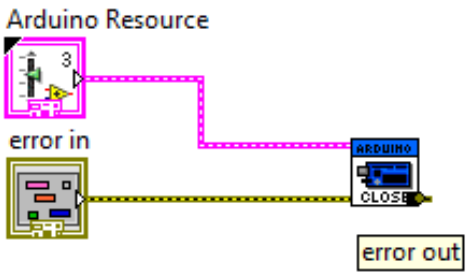


Figura 56 Entradas y salidas SubVI Close.vi

Una vez conocemos los subVI utilizados para iniciar y terminar la comunicación con LabVIEW, necesitamos otra serie de subVI para la obtención de los datos que nos

proporcionan los sensores y otros para la comunicación con los pines digitales que hayamos definido como de salida en arduino.

En nuestro proyecto vamos a utilizar los subVI de lectura Digital Write Pin (ver Figura 57 y 58), Analog Read Pin (ver Figura 59 y 60), Thermistor Read (ver Figuras 61 y 62) y Photocell Read (Ver Figura 63 y 64).

1. Digital Write Pin.vi

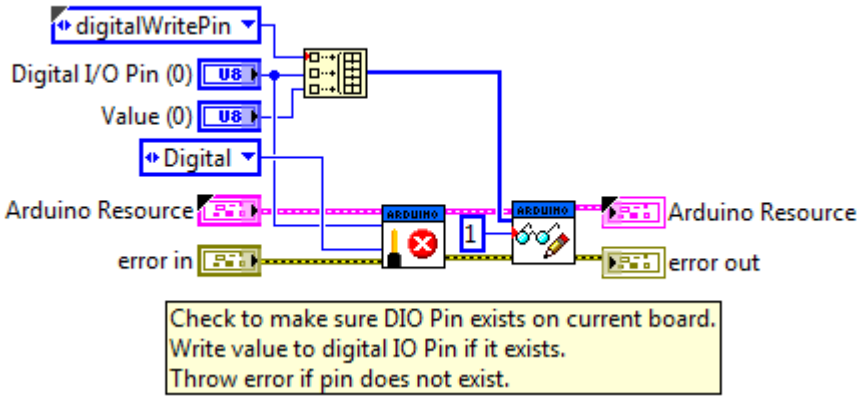


Figura 57 Diagrama de bloques Digital Write Pin.vi

Este subVI (ver Figura 57) lo que hace es escribir los pines digitales de salida con el valor HIGH o LOW. Cuenta con 4 entradas y 2 salidas (ver Figura 58) que vemos a continuación:

- Arduino Resource: Salida de información obtenida de Init.vi.
- Digital I/O Pin (0): Numero de pin digital en el cual se desea escribir. Viene con el valor default 0.
- Value (0): El valor que se desea escribir. Puede ser 0 (LOW) o 1 (HIGH). Viene por defecto con el valor 0.
- Error in: Es la entrada de errores.
- Arduino Resource 2: Salida de la línea de información obtenida en Init.vi.
- Error out: La salida de errores.

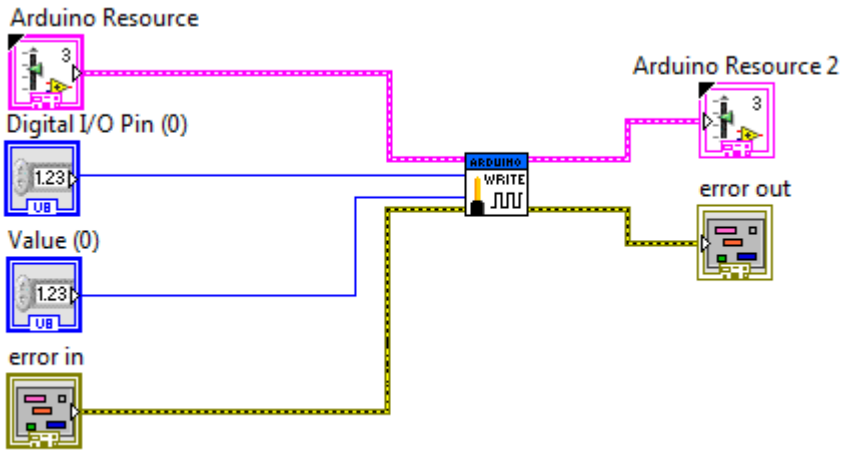


Figura 58 Entradas y salidas SubVI Digital Write Pin.vi

2. Analog Read Pin.vi

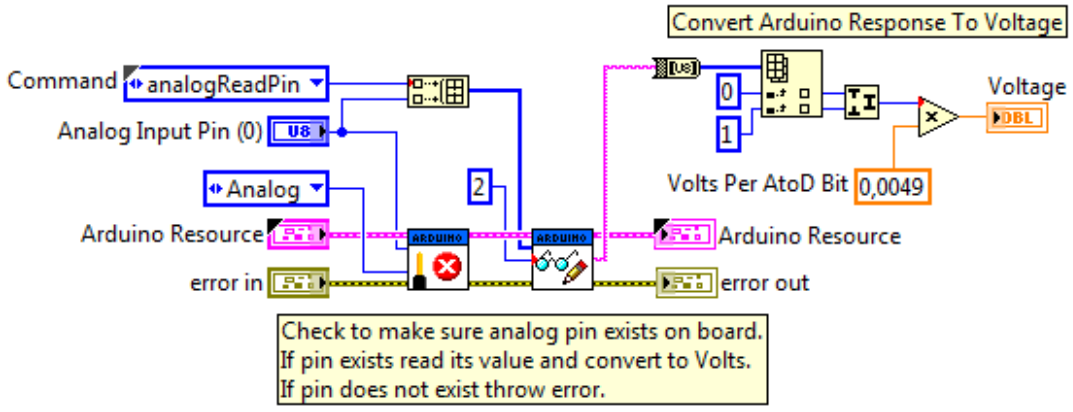


Figura 59 Diagrama de bloques Analog Read Pin.vi

Este subVI (ver Figura 59) lo que hace es leer el voltaje recibido en los pines analógicos de la placa arduino. Cuenta con 3 entradas y 3 salidas (ver Figura 60) que vemos a continuación:

- Arduino Resource: Salida de información obtenida de Init.vi.
- Analog Input Pin (0): Numero de pin analógico del cual se desea obtener el valor. Viene con el valor default 0.
- Error in: Es la entrada de errores.
- Arduino Resource 2: Salida de la línea de información obtenida en Init.vi.

- Voltaje: Valor del voltaje recibido en el pin analógico.
- Error out: La salida de errores.

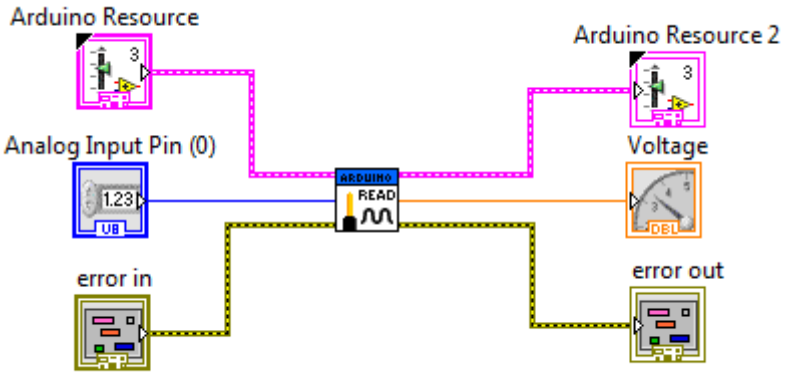


Figura 60 Entradas y salidas SubVI Analog Read Pin.vi

3. Thermistor Read.vi

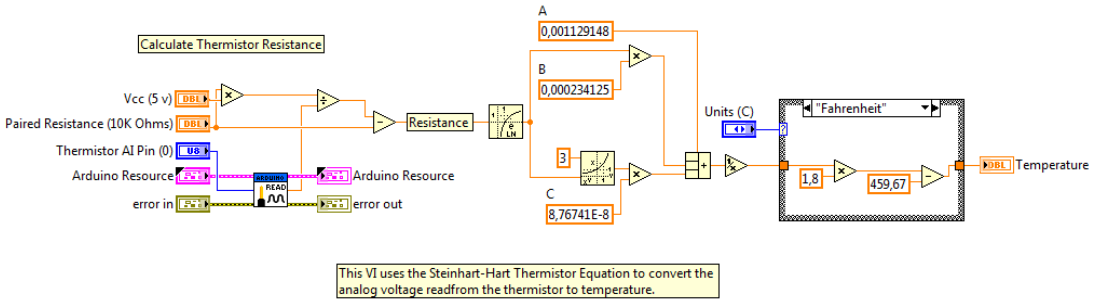


Figura 61 Diagrama de bloques Thermistor Read.vi

Este subVI (ver Figura 61) está especialmente diseñado para la medición de la temperatura por medio un termistor. Como se puede observar, está formado por un subVI Analog Read.vi y luego una serie de operaciones matemáticas que nos convierte el valor del voltaje de ese pin analógico en el valor de la temperatura medida. Este subVI cuenta con 6 entradas donde definiremos las condiciones del termistor, y 3 salidas en las que obtendremos los datos (ver Figura 62), que vemos a continuación:

- Arduino Resource: Salida de información obtenida de Init.vi.
- Thermistor AI Pin (0): Numero de pin analógico en el cual está conectado el termistor.
- Paired Resistance (10K Ohm): Resistencia asociada al termistor. Viene con un valor predeterminado de 10K ohm. En nuestro caso será de 5K ohm.
- Vcc (5v): Tensión de entrada al Thermistor. Viene con un valor predeterminado de 5 voltios. En nuestro caso será de 5 voltios.
- Error in: Es la entrada de errores.
- Units (C): Indicamos la unidad en que le queremos recibir el valor. Viene predeterminado en Celsius. Sera la que utilizaremos nosotros.
- Arduino Resource 2: Salida de la línea de información obtenida en Init.vi.
- Temperature: Nos da el valor de la temperatura asociado al valor leído en el termistor.
- Error out: La salida de errores.

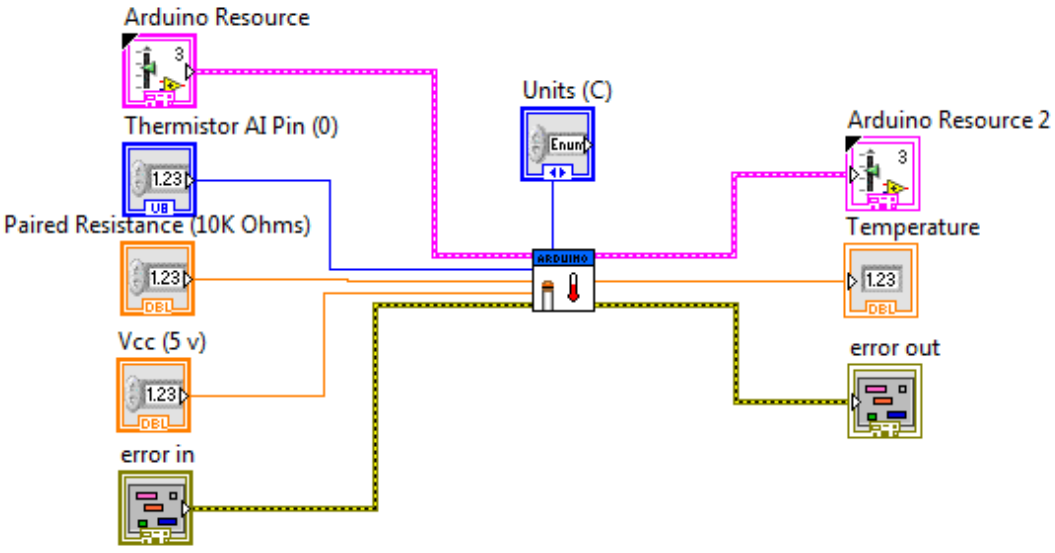


Figura 62 Entradas y salidas SubVI Thermistor Read.vi

4. Photocell Read.vi

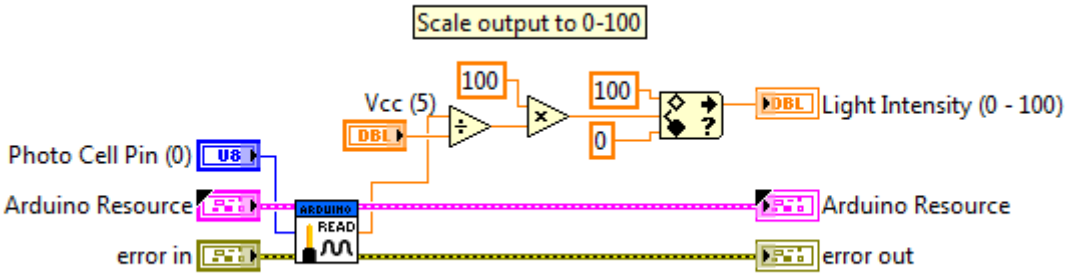


Figura 63 Diagrama de bloques Photocell Read.vi

Este subVI (ver Figura 63) está especialmente diseñado para la medición de la luminosidad por medio una fotorresistencia. Como se puede observar, está formado por un subVI Analog Read.vi y luego una serie de operaciones matemáticas que nos convierte el valor del voltaje de ese pin analógico en el valor de la luminosidad. Este subVI cuenta con 4 entradas donde definiremos las condiciones de la fotorresistencia, y 3 salidas en las que obtendremos los datos (ver Figura 64), que vemos a continuación:

- Arduino Resource: Salida de información obtenida de Init.vi.
- Photo Cell Pin (0): Numero de pin analógico en el cual está conectada la fotorresistencia.
- Vcc (5v): Tensión de entrada a la fotorresistencia. Viene con un valor predeterminado de 5 voltios. En nuestro caso será de 5 voltios.
- Error in: Es la entrada de errores.
- Arduino Resource 2: Salida de la línea de información obtenida en Init.vi.
- Light Intensity (0 - 100): Nos da el valor de la luminosidad asociada al valor leído en la fotorresistencia. Está en una escala porcentual.
- Error out: La salida de errores.

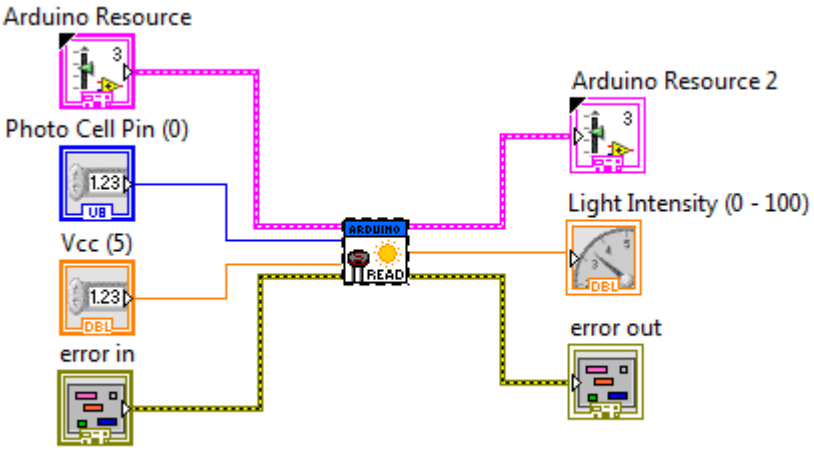


Figura 64 Entradas y salidas SubVI Photocell Read.vi

3. *VI Principal*

En la siguiente figura 65 podemos ver el panel frontal que se ha diseñado para este proyecto que nos permite visualizar las medidas una vez conectada la placa arduino. Si observamos la imagen del panel frontal del programa encontramos 4 partes diferenciadas. Una destinada a salida de datos Excel, otra a luminosidad, otra destinada a la humedad del suelo y una última parte destinada a la temperatura.

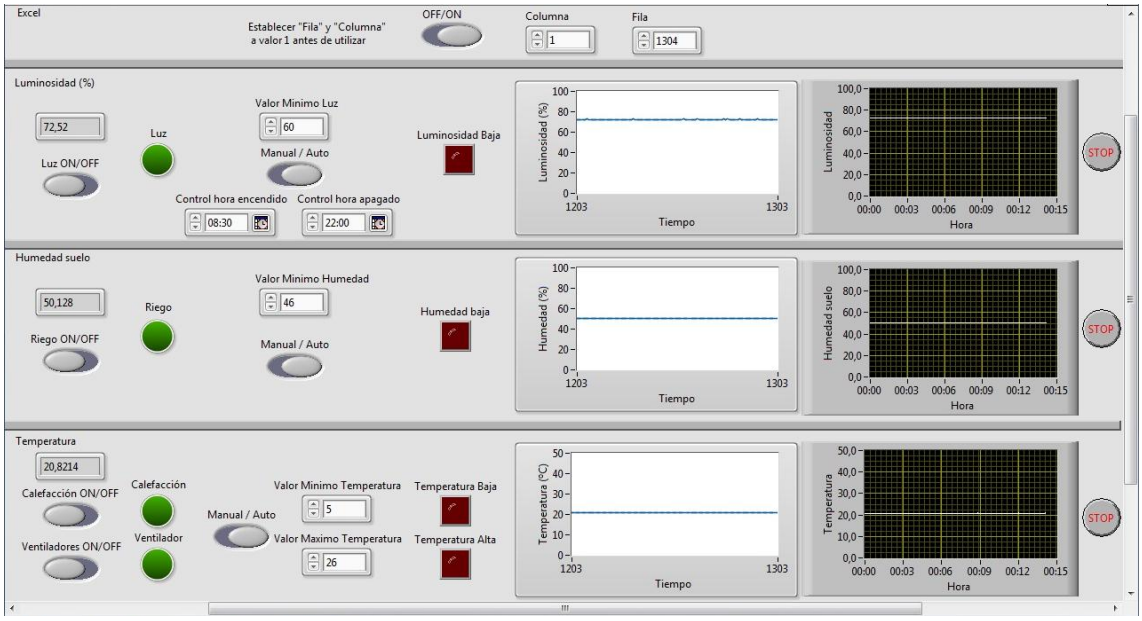


Figura 65 Panel frontal LabVIEW

A continuación podemos observar el diagrama de bloques asociado a este vi. Debido a su tamaño y funcionalidad, se va a dividir en 2 grandes bloques. En una primera parte (ver Figura 66) se encuentra el inicio del programa y la primera parte del flat secuence. Y una segunda parte (ver Figura 67) se encuentra la parte correspondiente a la segunda parte del flat secuence y el cierre del programa.

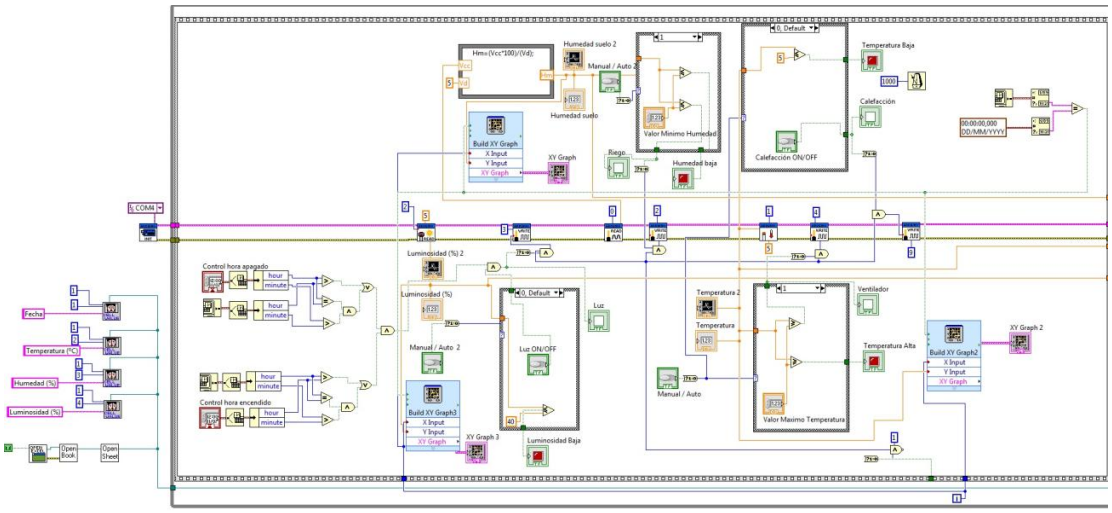


Figura 66 Diagrama de bloques principal: Primera parte

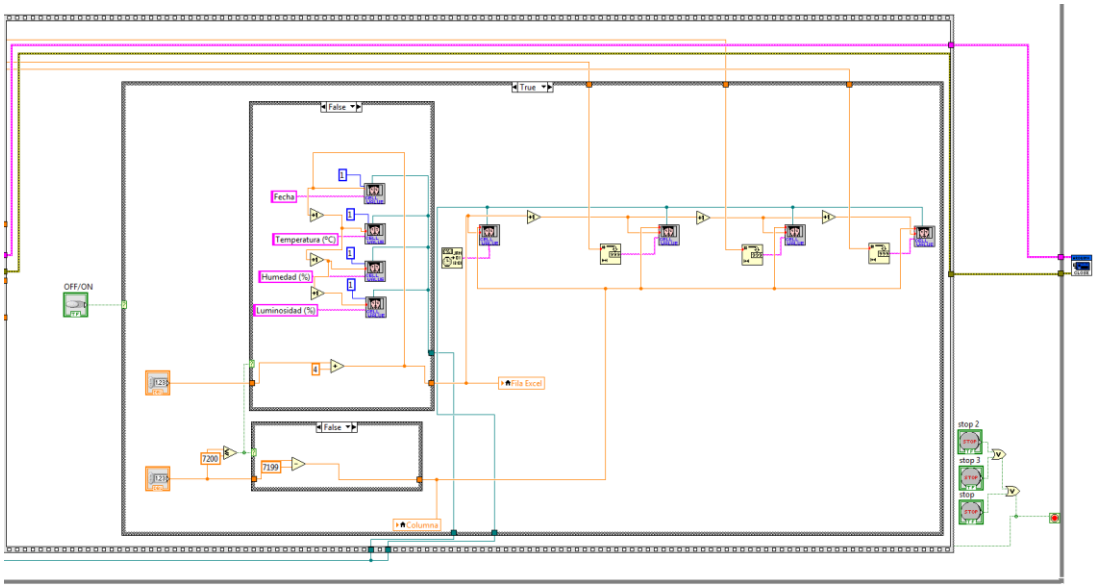


Figura 67 Diagrama de bloques principal: Segunda parte

Esta sistema no se detendrá hasta que se active el botón “ Stop” de la estructura While Loop que incorpora el botón de control para parar el bucle esta estructura nos incluye el botón stop en el panel de controles en la figura podemos ver su apariencia

en el diagrama de bloques. A continuación explicaremos con más detalles los pasos seguidos.

1. Inicio del programa

El inicio del programa se considera a todo aquello que se ejecuta antes de la inicialización del while loop. En este apartado podemos observar por un lado la función Init.vi para el arranque de la comunicación con arduino y por otro, la apertura de Excel, y la escritura de la primera fila (la exportación de datos se verá en el Capítulo 6)

2. Control y configuración de la luminosidad

En este punto vamos a estudiar tanto el diagrama de bloques como el panel frontal asociado a la luminosidad.

El panel frontal (ver Figura 68) cuenta con un indicador numérico de la luminosidad, un control numérico para el valor mínimo de luz, dos controles de hora para establecer la hora mínima y máxima de encendido, dos interruptores (uno para la selección de funcionamiento entre manual y automático y otro dentro del funcionamiento manual para la activación del relé asociado a la luz) así como un led que nos indica el funcionamiento de la luz, una alarma que nos indica si el valor de la luminosidad es menor que el designado o predeterminado, dos graficas las cuales muestran la evolución temporal de la luminosidad y por ultimo un botón de STOP para realizar la parada del programa.

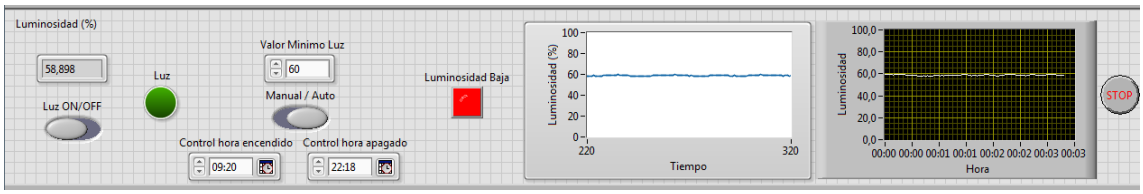


Figura 68 Panel frontal luminosidad

La primera de las gráficas nos muestra la información referente a los últimos 100 segundos, mientras que la segunda de las gráficas muestra la evolución de la luminosidad durante toda la ejecución del programa.

El diagrama de bloques correspondiente a la luminosidad (ver Figura 69), se encuentra englobado dentro del primer frame del flat secuence.

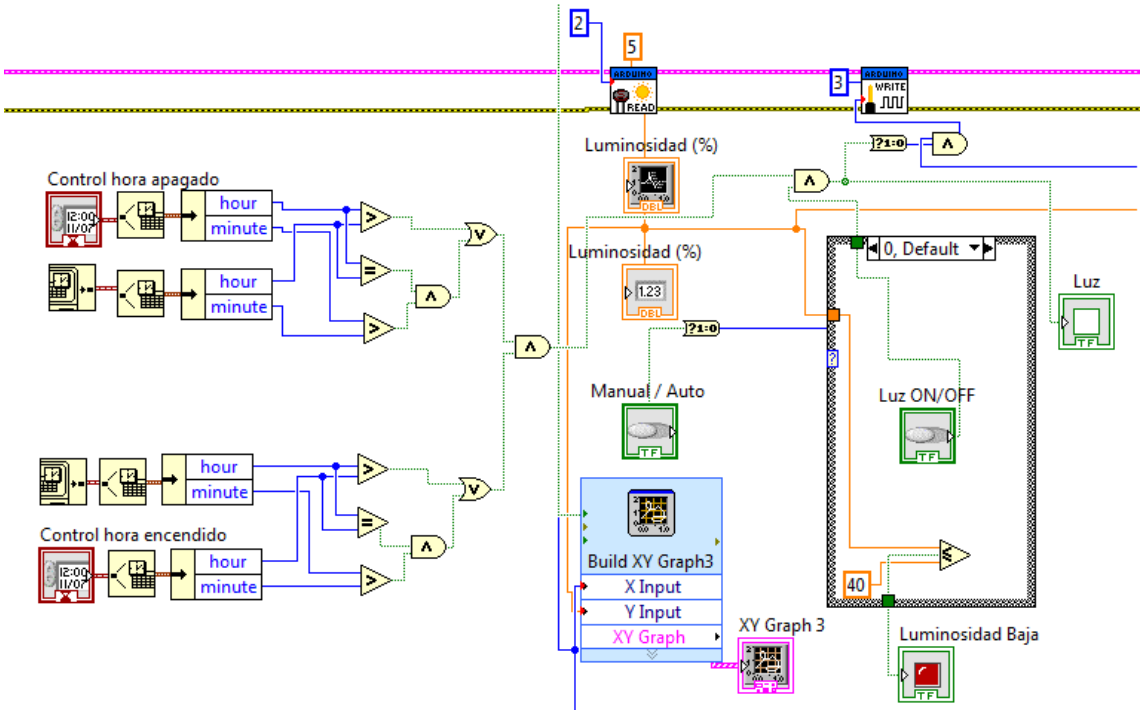


Figura 69 Diagrama de bloques luminosidad

En este diagrama de bloques se puede observar que comienza con la lectura de la fotorresistencia, situada en el pin analógico 2 y conectada a 5 voltios. La salida de los datos procedente del módulo Photocell Read se conecta a el grafico Luminosidad (%) , al indicador numérico del mismo nombre, a la configuración del XY Graph3, y al case structure.

El case structure tiene dos posiciones controladas por el interruptor Manual/Auto. La posición 0 o default correspondiente a la posición manual, y la posición 1 corresponde a la posición Automática (ver Figura 70). El case structure cuenta a su vez con dos salidas. Una correspondiente a la alarma y otra correspondiente al control del relé asociado a la luz.

Por último se encuentra el modulo Digital Write.vi el cual está asociado al pin digital 3, y en cuya entrada se encuentra una puerta AND. Esta puerta AND cuenta a su entrada con una puerta AND cuyas entradas son la salida del case structure y la salida del control de horarios, y otra entrada correspondiente al botón STOP, de forma que cuando se active el botón, se apague el relé automáticamente.

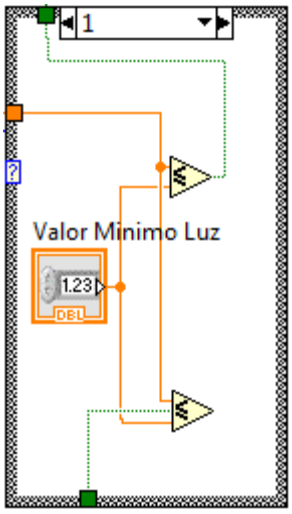


Figura 70 Posición 1 case structure luminosidad.

3. Control y configuración de la humedad del suelo

En este punto vamos a estudiar tanto el diagrama de bloques como el panel frontal asociado a la humedad relativa. Es un bloque idéntico al de luminosidad

El panel frontal (ver Figura 71) cuenta con un indicador numérico de la humedad del suelo, un control numérico para el valor mínimo de luz, dos interruptores (uno para la selección de funcionamiento entre manual y automático y otro dentro del funcionamiento manual para la activación del relé asociado al riego) así como un led que nos indica el funcionamiento del riego, una alarma que nos indica si el valor de la humedad del suelo es menor que el designado o predeterminado, dos graficas las cuales muestran la evolución temporal de la humedad del suelo y por ultimo un botón de STOP para realizar la parada del programa.

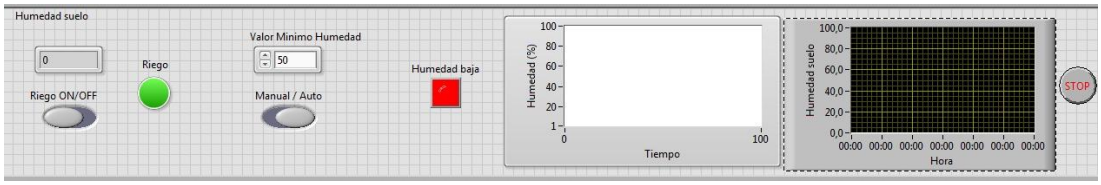


Figura 71 Panel frontal humedad del suelo

La primera de las gráficas nos muestra la información referente a los últimos 100 segundos, mientras que la segunda de las gráficas muestra la evolución de la humedad del suelo durante toda la ejecución del programa.

El diagrama de bloques correspondiente a la humedad del suelo (ver Figura 72), se encuentra englobado dentro del primer frame del flat secuence.

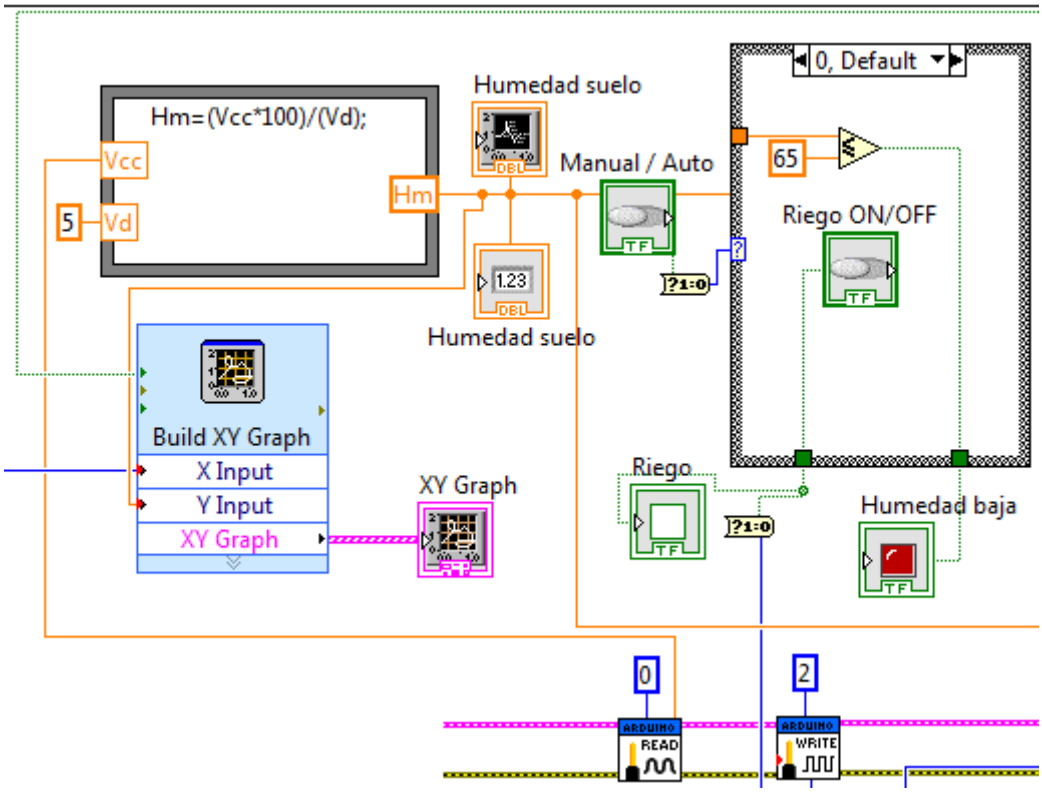


Figura 72 Diagrama de bloques humedad del suelo

En este diagrama de bloques se puede observar que comienza con la lectura de la fotorresistencia, situada en el pin analógico 0. La salida de los datos procedente del subVI Analog Read se conecta a un formula node. En este formula node se convierte el valor del voltaje obtenido del Analog Read en el valor de la humedad del suelo. La salida del formula node se conecta al grafico Humedad suelo, al indicador numérico del mismo nombre, a la configuración del XY Graph, y al case structure.

El case structure tiene dos posiciones controladas por el interruptor Manual/Auto. La posición 0 o default correspondiente a la posición manual, y la posición 1 corresponde a la posición Automática (ver Figura 73). El case structure cuenta a su vez con dos salidas. Una correspondiente a la alarma y otra correspondiente al control del relé asociado al riego.

Por último se encuentra el modulo Digital Write.vi el cual está asociado al pin digital 2, y en cuya entrada se encuentra una puerta AND, con una entrada correspondiente a la salida del case structure y otra entrada correspondiente al botón STOP, de forma que cuando se active el botón, se apague el relé automáticamente.

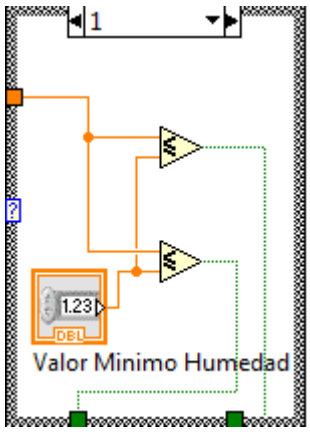


Figura 73 Posición 1 case structure humedad suelo

4. Control y configuración de la temperatura

En este punto vamos a estudiar tanto el diagrama de bloques como el panel frontal asociado a la temperatura.

El panel frontal (ver Figura 74) cuenta con un indicador numérico de la humedad del suelo, dos controles numéricos para el valor mínimo y máximo de la temperatura, tres interruptores (uno para la selección de funcionamiento entre manual y automático y dos dentro del funcionamiento manual para la activación de los relés asociado a los ventiladores y la calefacción) así como un dos led que nos indican el funcionamiento de los ventiladores y calefacción, dos alarmas que nos indican si el valor de la temperatura es menor o mayor que el designado o predeterminado, dos graficas las cuales muestran la evolución temporal de la temperatura y por ultimo un botón de STOP para realizar la parada del programa.

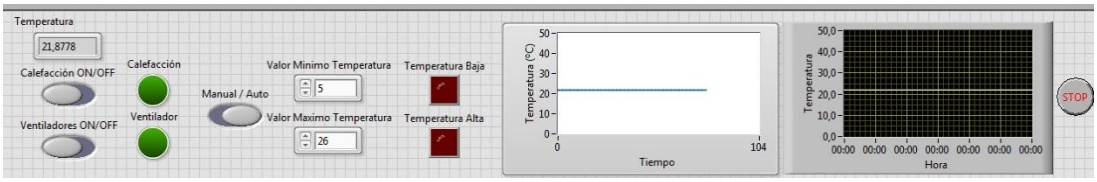


Figura 74 Panel frontal temperatura

La primera de las gráficas nos muestra la información referente a los últimos 100 segundos, mientras que la segunda de las gráficas muestra la evolución de la temperatura durante toda la ejecución del programa.

El diagrama de bloques correspondiente a la temperatura (ver Figura 75), se encuentra englobado dentro del primer frame del flat secuence.

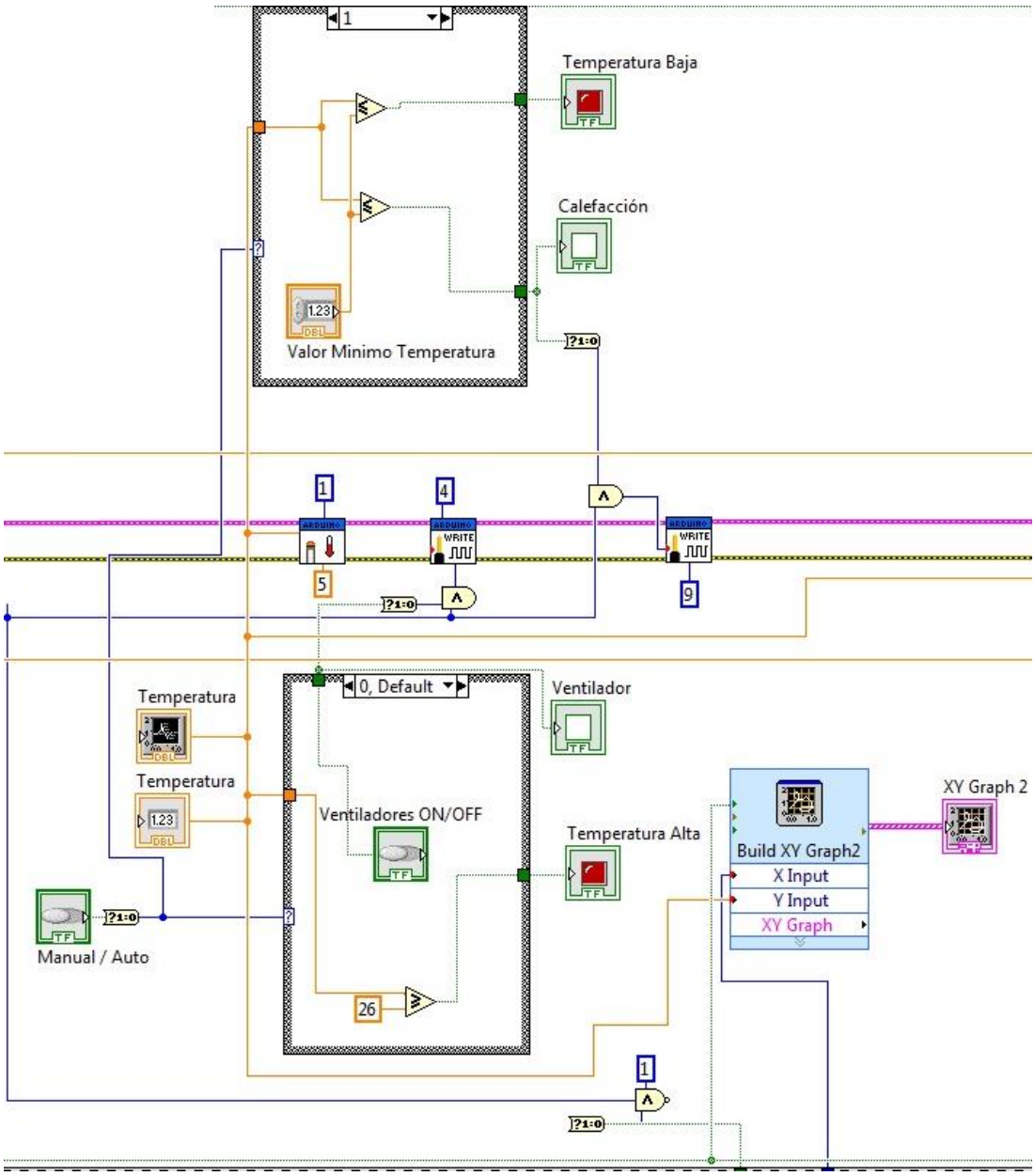


Figura 75 Diagrama de bloques temperatura

En este diagrama de bloques se puede observar que comienza con la lectura del Thermistor Read.vi, situada en el pin analógico 1. La salida de los datos procedente del subVI Thermistor Read se conecta al grafico Temperatura, al indicador numérico del mismo nombre, a la configuración del XY Graph2, y a los dos case structure destinados a ventiladores y calefacción respectivamente.

El case structure destinado a los ventiladores tiene dos posiciones controladas por el interruptor Manual/Auto. La posición 0 o default correspondiente a la posición manual, y la posición 1 corresponde a la posición Automática (ver Figura 76). El

case structure cuenta a su vez con dos salidas. Una correspondiente a la alarma y otra correspondiente al control del relé asociado a los ventiladores.

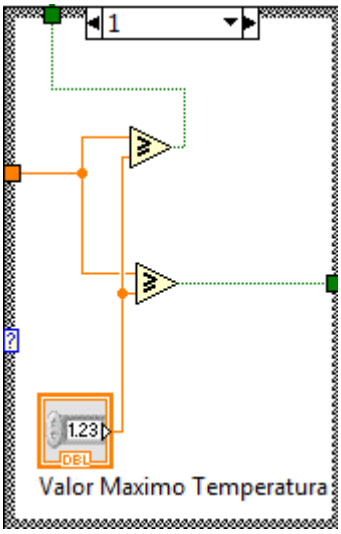


Figura 76 Posición 1 case structure ventiladores

El case structure destinado a la calefacción tiene dos posiciones controladas por el interruptor Manual/Auto. La posición 0 o default correspondiente a la posición manual (ver Figura 77), y la posición 1 corresponde a la posición Automática. El case structure cuenta a su vez con dos salidas. Una correspondiente a la alarma y otra correspondiente al control del relé asociado a la calefacción.

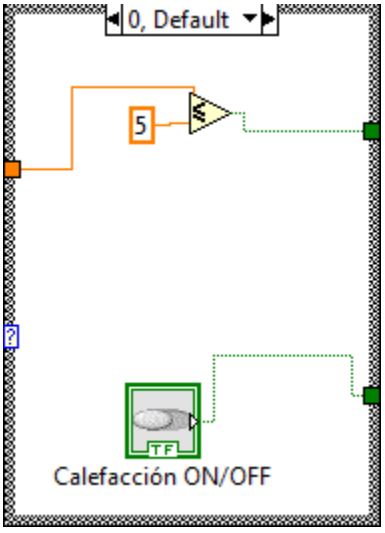


Figura 77 Posición 0 case structure calefacción

Por último se encuentran los módulos Digital Write.vi los cuales están asociados al pin digital 4 para el caso de los ventiladores y 9 para el caso de la calefacción, y en cuyas entradas se encuentra una puerta AND, con una entrada correspondiente a la salida del case structure y otra entrada correspondiente al botón STOP, de forma que cuando se active el botón, se apague el relé automáticamente.

5. Configuración tiempo de muestreo

En este punto se expone la estructura auxiliar necesaria para establecer un tiempo de muestro, que en nuestro caso se establece en 1 segundo. Para ello es necesaria la colocación de un "Wait Until Next ms Multiple" estableciendo su entrada con una constante de valor 1000 (ver Figura 78). Esto colocado dentro de un frame y sin conectarlo a ningún elemento, producirá que espere un segundo para cada ejecución de este frame.

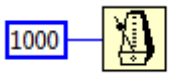


Figura 78 "Wait Until Next ms Multiple" 1000 ms

6. Finalización del programa

La finalización del programa se realiza mediante la pulsación de cualquiera de los botones STOP colocados en el panel frontal.

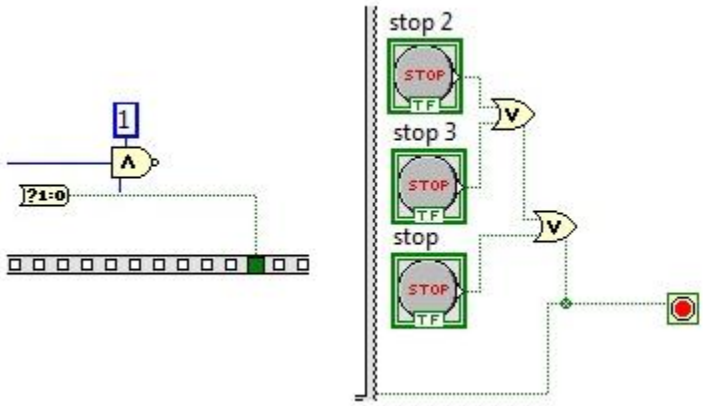


Figura 79 Diagrama bloques STOP

Como se puede observar en la Figura 79, con la pulsación de cualquiera de los botones STOP se activa el stop del While Loop. Además por medio de la puerta NAND, se envía el valor 0 a la puerta AND conectada a la entrada de cada subVI Digital Write Pin.vi, lo que produce el apagado automático de los relés.

Una vez detenido el While Loop se ejecuta la subVI Close.VI y se cierra la comunicación con la placa arduino.

CAPÍTULO 6. Exportar datos a Excel

1. Introducción

En este capítulo se describe el proceso de una función típica en LabVIEW como sería la exportación de datos generados en LabVIEW directamente a Microsoft Excel.

2. Programación LabVIEW

La programación necesaria para la exportación de datos desde LabVIEW a Microsoft Excel sigue una serie de pasos que serán los siguientes:

- Abrir Microsoft Excel
- Abrir libro de trabajo
- Abrir una hoja de cálculo
- Agregar encabezados y valores a la hoja de calculo

A continuación se pasa a exponer cada uno de los pasos citados.

1. Abrir Microsoft Excel

En este punto se expone como abrir el programa Microsoft Excel desde LabVIEW. Para ello será necesario la utilización del subVI “Open Excel and Make Visible.vi” (ver Figura 80). Este subVI lo que hace es llamar a una aplicación externa (Microsoft Excel) y abrirla.

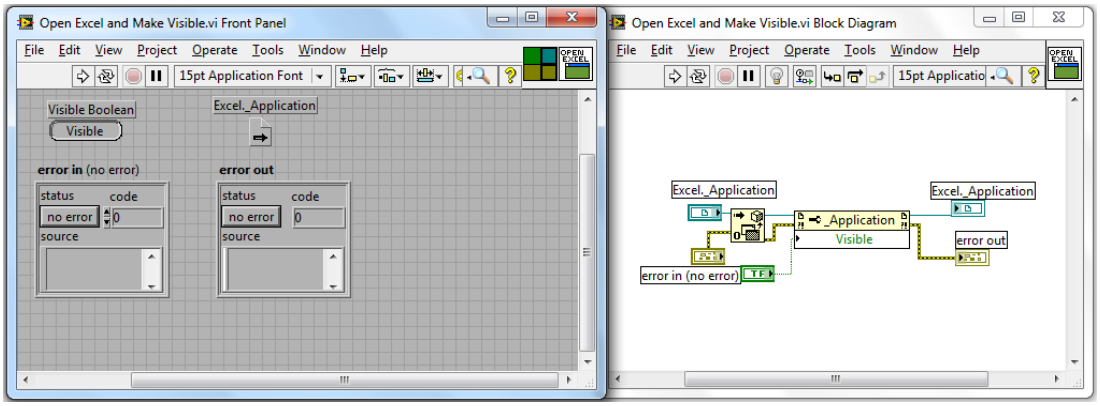


Figura 80 Open Excel and Make Visible.vi

2. *Abrir libro de trabajo*

Una vez abierta la aplicación Microsoft Excel, es necesario la apertura de un nuevo libro de trajo. Para ello utilizaremos el subVI "Open New WorkBook.vi" (ver Figura 81).

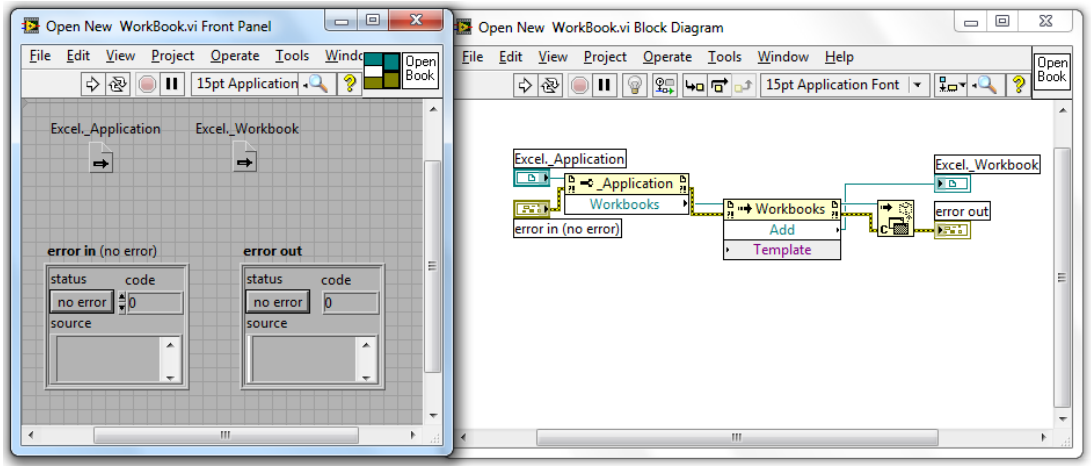


Figura 81 Open New WorkBook.vi

Esta subVI lo que hace es, una vez abierta la aplicación Microsoft Excel, nos crea un nuevo WorkBook en el cual vamos a trabajar. Para ello sigue una serie de pasos que son:

- Crea una referencia a un WorkBook
- Una vez tiene la referencia del libro (refnum), crea un nuevo libro.

- Cierra la referencia del viejo WorkBook y conserva la referencia del nuevo WorkBook.

3. *Abrir una hoja de cálculo*

Una vez ya tenemos abierto Microsoft Excel y tenemos creado un nuevo WorkBook, será necesaria la apertura de una nueva hoja de cálculo dentro de este WorkBook. Para ello será necesario la introducción de otro un subVI que pueda realizar esta tarea.

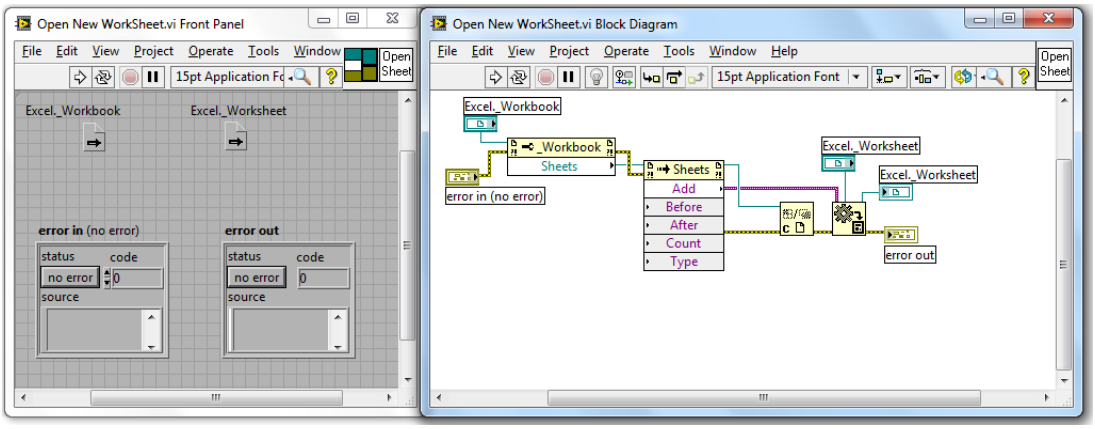


Figura 82 Open New WorkSheet.vi

La subVI necesaria en este caso es "Open New WorkSheet.vi" (ver Figura 82). Esta subVI sigue unos pasos que se exponen a continuación:

- Obtiene el refnum de Sheets de una de las propiedades de objeto WorkBook.
- Crea una nueva Sheet.
- Cierra el refnum de la propiedad del libro.
- Convierte la hoja de Excel en una variable entendible por LabVIEW.

4. *Agregar encabezados y valores a la hoja de cálculo*

En este punto se explica cómo insertar los valores y encabezados en la hoja de cálculo de Microsoft Excel abierto en los pasos anteriores. Para ello se va utilizar una nueva subVI llamada "Set Cell Value.vi" (ver Figura 83).

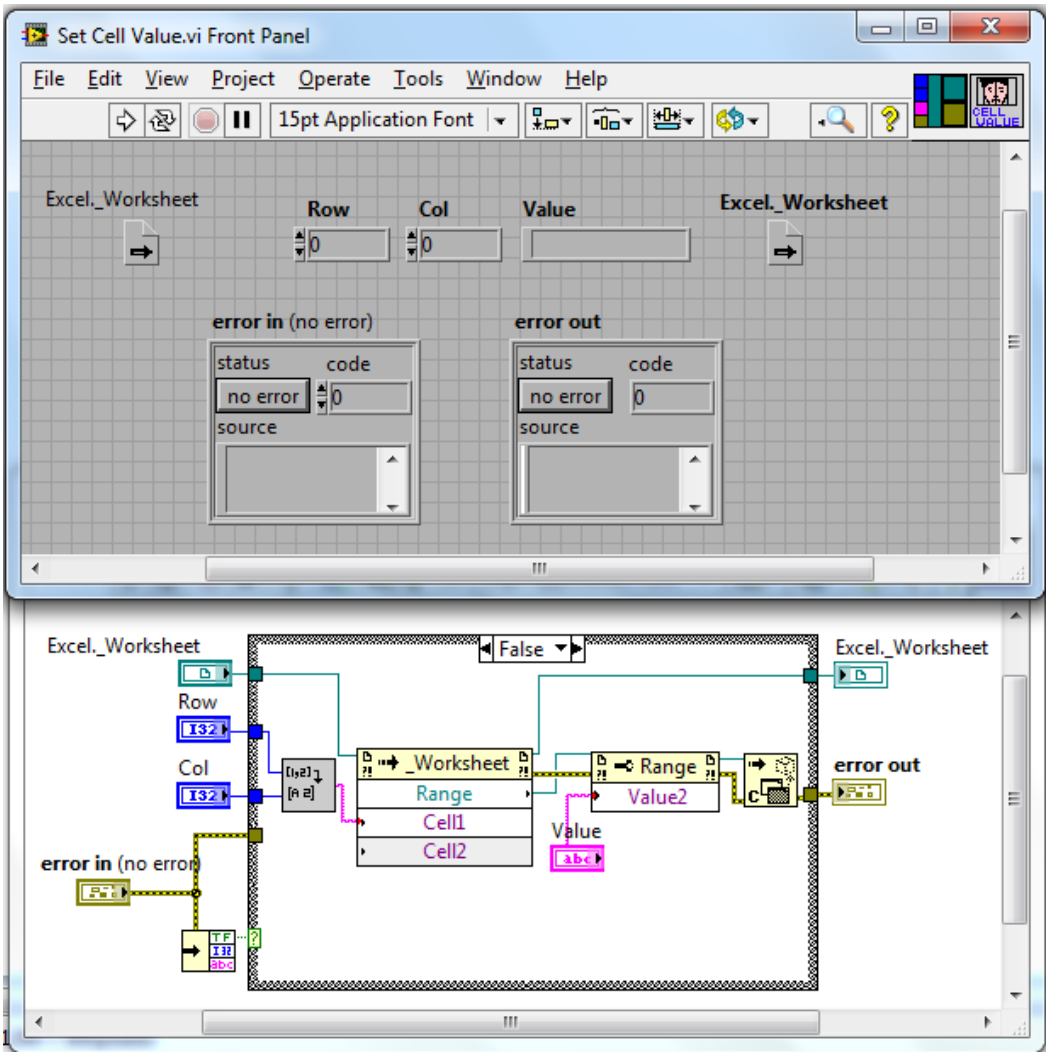


Figura 83 Set Cell Value.vi

Este subVI se encarga de introducir un valor en una celda de la hoja de cálculo. Recibe como entradas los la fila y columna donde se quiere insertar, así como el valor que se ha de insertar en la misma.

A través del subVI "Row Col To Range Format.vi" (ver Figura 84), se encarga de localizar la celda que corresponde dentro de Microsoft Excel y le transfiere el valor a dicha celda.

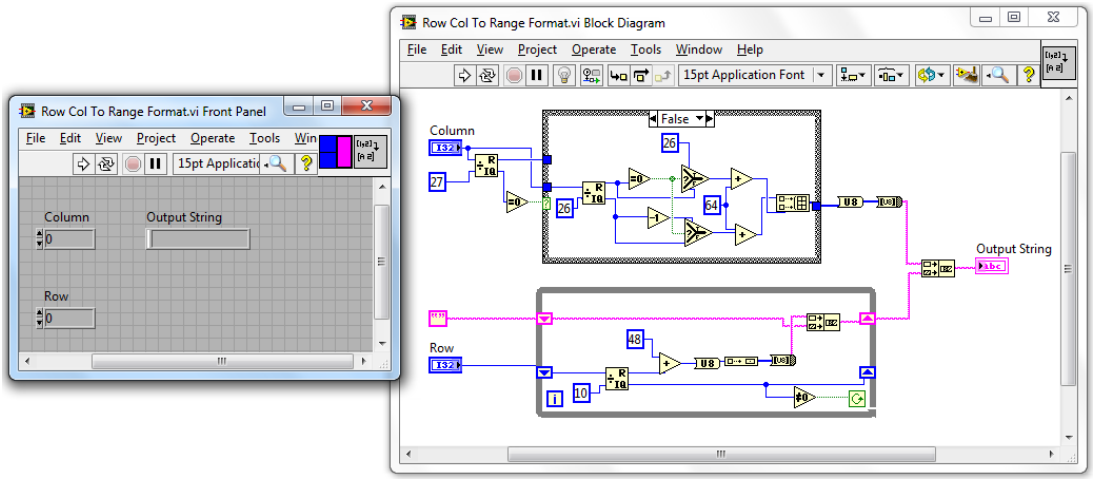


Figura 84 Row Col To Range Format.vi

Este VI se divide en dos partes. Primeramente la obtención de la fila y en segundo lugar la obtención de la columna. La fila no presenta un problema al ser un valor numérico, se transforma el valor de la fila a bytes, construye una matriz con ellos y seguidamente lo vuelve a pasar a numérico y lo junta con una constante en blanco mediante un bundle para inicializar la fila en blanco. Las columnas sí que presentan problemas. En Excel las columnas se indican alfabéticamente, y esta función se encarga de transformar el valor numérico a su valor alfabético equivalente y como en el caso de las filas se transforma a binario y se pasa a matriz. Finalmente los dos valores se juntan mediante un bundle y se recibe el valor final de la celda que se va a modificar en "output string".

Una se tiene localizada la celda donde se va introducir los valores, solo queda introducir los mismo. Se va a comenzar con la introducción de los encabezados y posteriormente se explicara cómo se introducen los valores de forma ordenada.

1. Introducción de encabezados

La introducción de encabezados se va a realizar en dos pasos diferenciados. El primero de ellos se encargara de introducir los primeros encabezados acto seguido de abrir la hoja de cálculo, y el segundo lo hará cada vez que alcancemos el número máximo de filas definido (en nuestro caso será 7201 correspondientes a 2 horas de muestro).

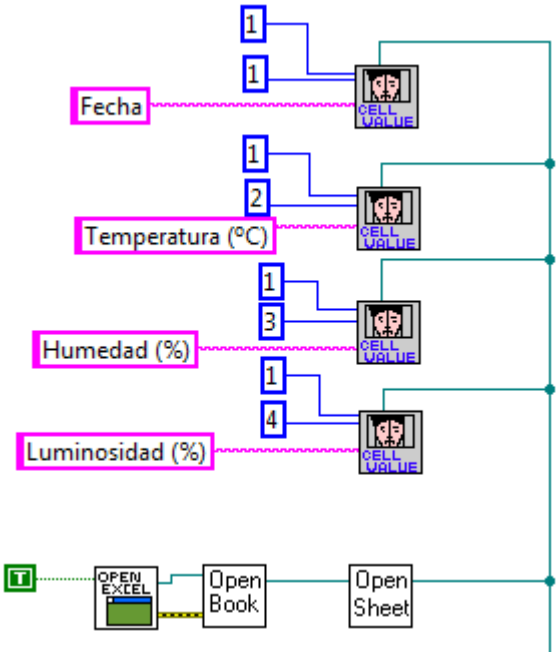


Figura 85 Diagrama de bloques introducción inicial de encabezados

Para introducir los primeros encabezados (ver Figura 85) se ejecuta el subVI "Set Cell Value.vi" para cada una de las primeras cuatro columnas, con el string asociado a cada celda. Esto se realiza fuera del While loop, al iniciar el programa y solo se ejecuta una vez quedando de la siguiente forma la hoja de cálculo de Microsoft Excel:

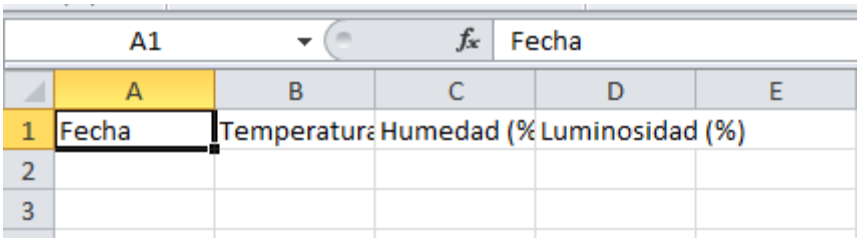


Figura 86 Hoja de cálculo Excel con encabezados

La introducción de los siguientes encabezados se realizara una vez alcanzado el número máximo de filas establecido. Esto se realiza dentro del segundo frame del flat secuence, que se encuentra dentro del While loop. Para ello como se puede observar en la figura 87 se declara un Case Structure. El Case Structure tiene como "?" la comparación del valor fila con la constante 7200, de forma que una vez alcanzado el valor 7201 se activan los case False. El primero de los Case Structure lo

que hace el volver a establecer el valor fila a 1, y el segundo de ellos incrementa el valor del contador columna en cuatro, e introduce los nuevos encabezados.

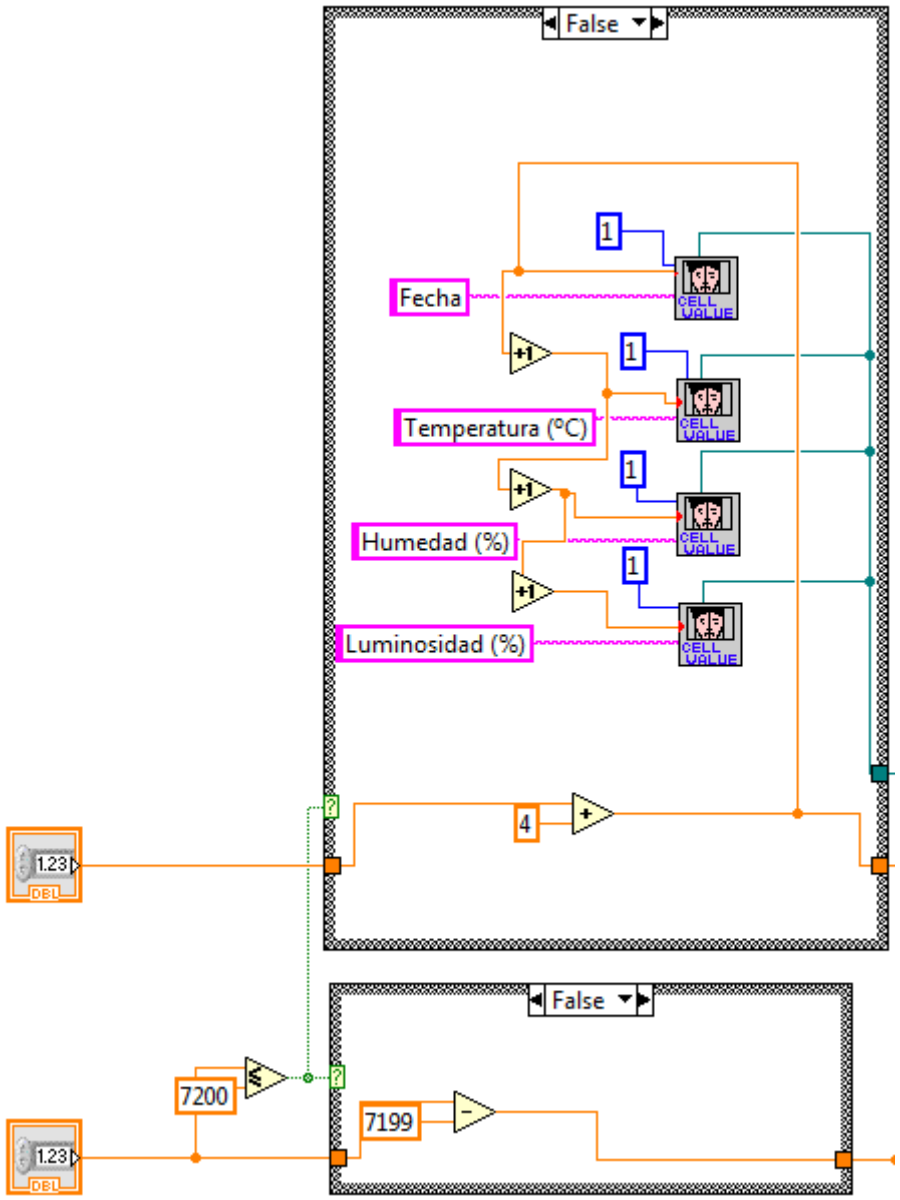


Figura 87 Diagrama de bloques introducción sucesivos encabezados

A continuación se muestra como queda la tabla Excel tras la introducción de los sucesivos encabezados:

	A	B	C	D	E	F	G	H
1	Fecha	Temperatura	Humedad (%)	Luminosidad (%)	Fecha	Temperatura	Humedad (%)	Luminosidad (%)
2	30/05/2012 15:01:21	24	52	73	30/05/2012 17:01:21	24	52	71
3	30/05/2012 15:01:22	24	52	73	30/05/2012 17:01:22	24	52	71
4	30/05/2012 15:01:23	24	52	73	30/05/2012 17:01:23	24	52	71
5	30/05/2012 15:01:24	24	52	73	30/05/2012 17:01:24	24	52	71
6	30/05/2012 15:01:25	24	52	73	30/05/2012 17:01:25	24	52	71
7	30/05/2012 15:01:26	24	52	73	30/05/2012 17:01:26	24	52	71
8	30/05/2012 15:01:27	24	52	73	30/05/2012 17:01:27	24	52	71

Figura 88 Hoja de cálculo Excel Varios encabezados

2. Introducción de valores

Una vez hemos introducido los encabezados, pasamos a la introducción de los datos. Esto se realizara en el segundo Frame del Flat Secuence. Se realizara por medio de "Set Cell Value". La posición se define por medio de los contadores de filas y columnas anteriormente vistos. Los datos se introducen en forma de string por medio del "Number To Decimal String", al cual alimentamos con los valores salidos de los sensores.

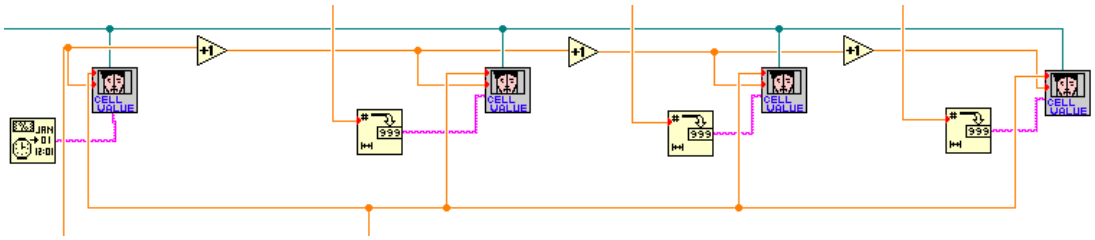


Figura 89 Diagrama de bloques para la introducción de datos en las celdas

La única excepción será el tiempo que se alimenta por medio del módulo "Format Date/Time String" que nos da la fecha y hora en formato string.

CAPÍTULO 7. Ejecución física del proyecto

1. Introducción

En este punto se va tratar la construcción física del sistema para el control y monitorización de un invernadero doméstico. Se expone por medio de imágenes la propuesta realizada en este proyecto.

2. Elementos utilizados

En la ejecución física de nuestro proyecto se han utilizado una serie de elementos además de los componentes anteriormente visto. En la tabla siguiente se muestra la lista de elementos adicionales utilizados:

Elemento	Cantidad
Portafusible	2
Fusible 2A	1
Fusible 3A	1
Interruptor	1
Conmutador	1
Regleta	1
Caja empalmes superficie estanca 160*100 mm	1
Caja empalmes redonda 70x36 mm	1
Cable rojo y negro 5m	1
Diodo LED verde	1
Resistencia 100Ω	1
Bridas, adhesivos doble cara y silicona fría	

Tabla 8 Elementos utilizados en la construcción del sistema

Estos elementos son fácilmente encontrados en cualquier tienda tanto de electricidad como en tiendas de bricolaje general.

3. Imágenes y esquemas

En este punto se muestran una serie de imágenes correspondientes a la instalación realizada, así como el esquema eléctrico de la conexión.

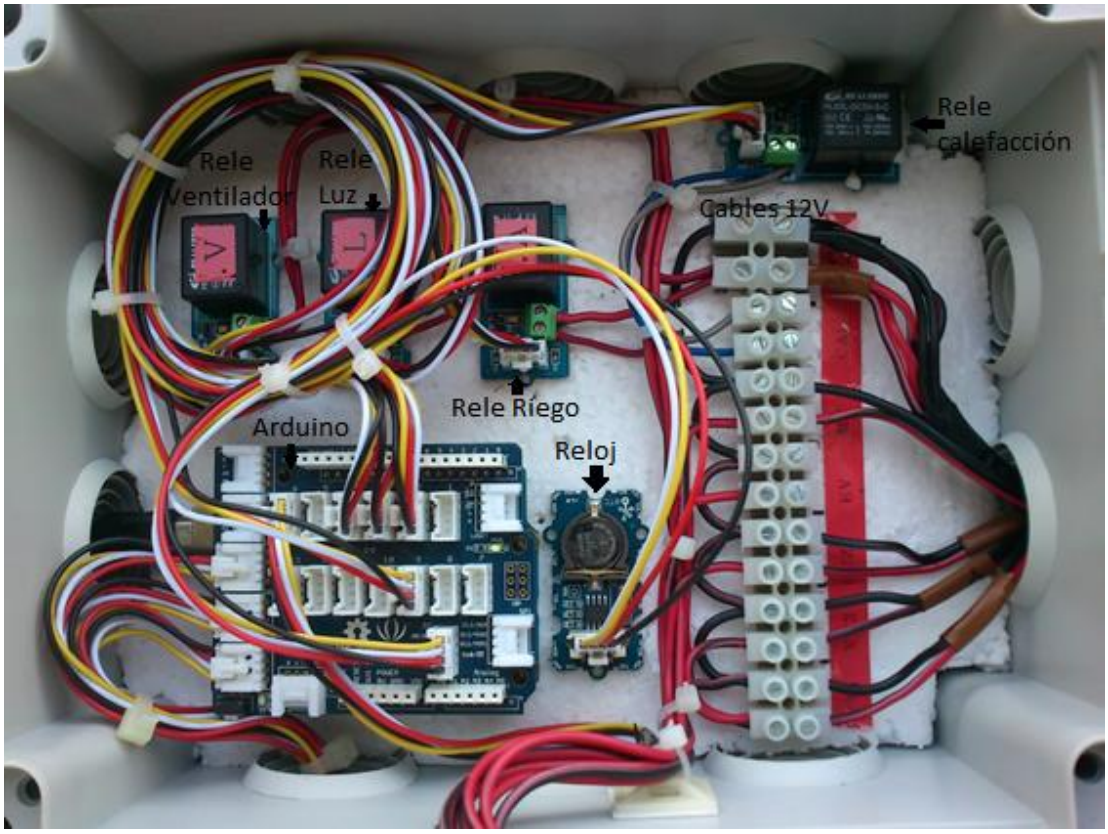


Figura 90 Interior caja instalación

La conexión de cables sigue el siguiente esquema, de forma que la conexión se realice a través de una regleta y no haya que manipular los relés.

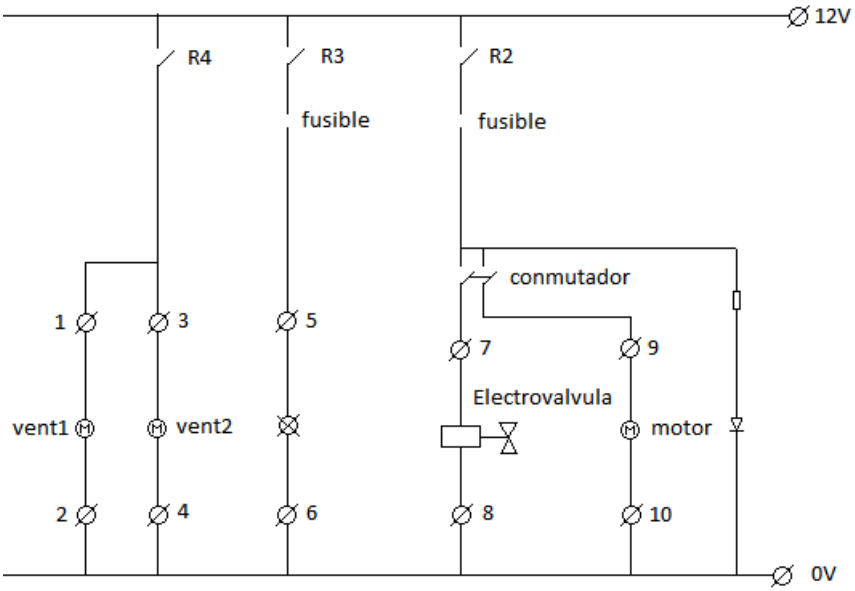


Figura 91 Esquema eléctrico 12V

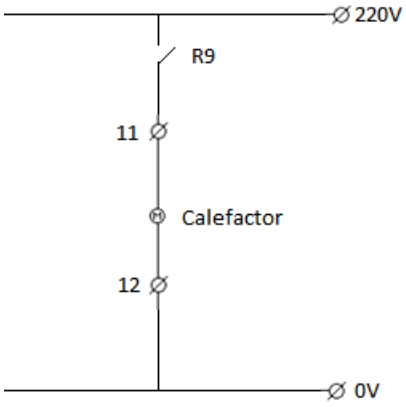


Figura 92 Esquema eléctrico 220V

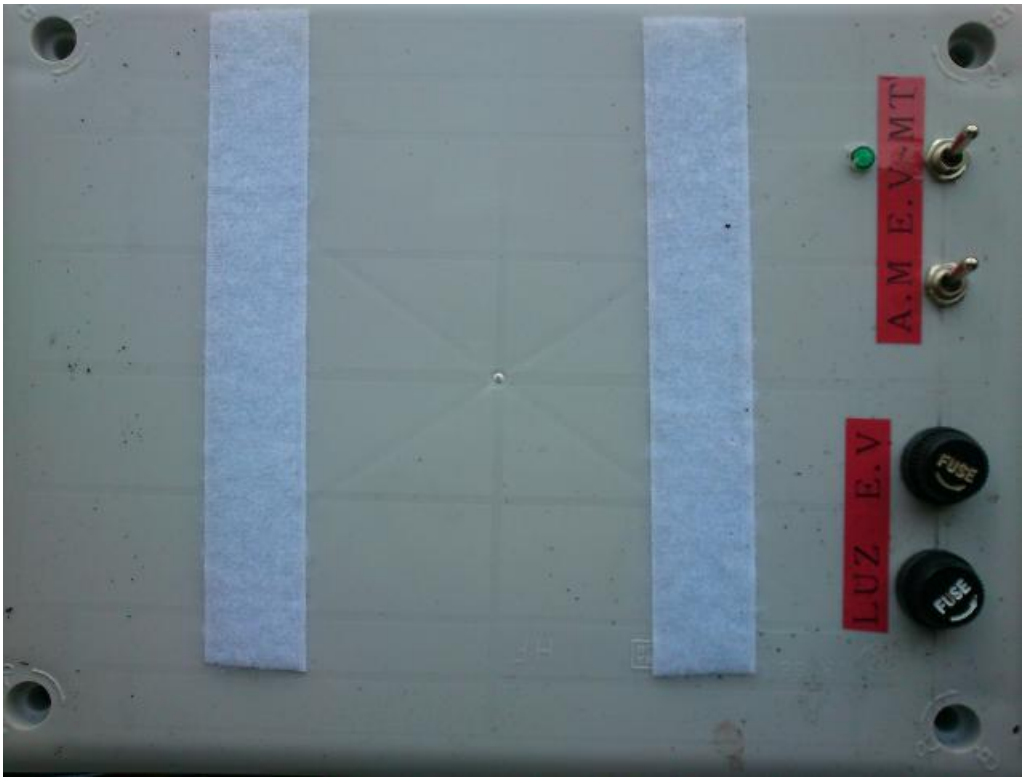


Figura 93 Tapa caja instalación

Figura 94 Interruptor, conmutador, fusibles y salida de cables corriente

Figura 95 Entrada conexión cable usb

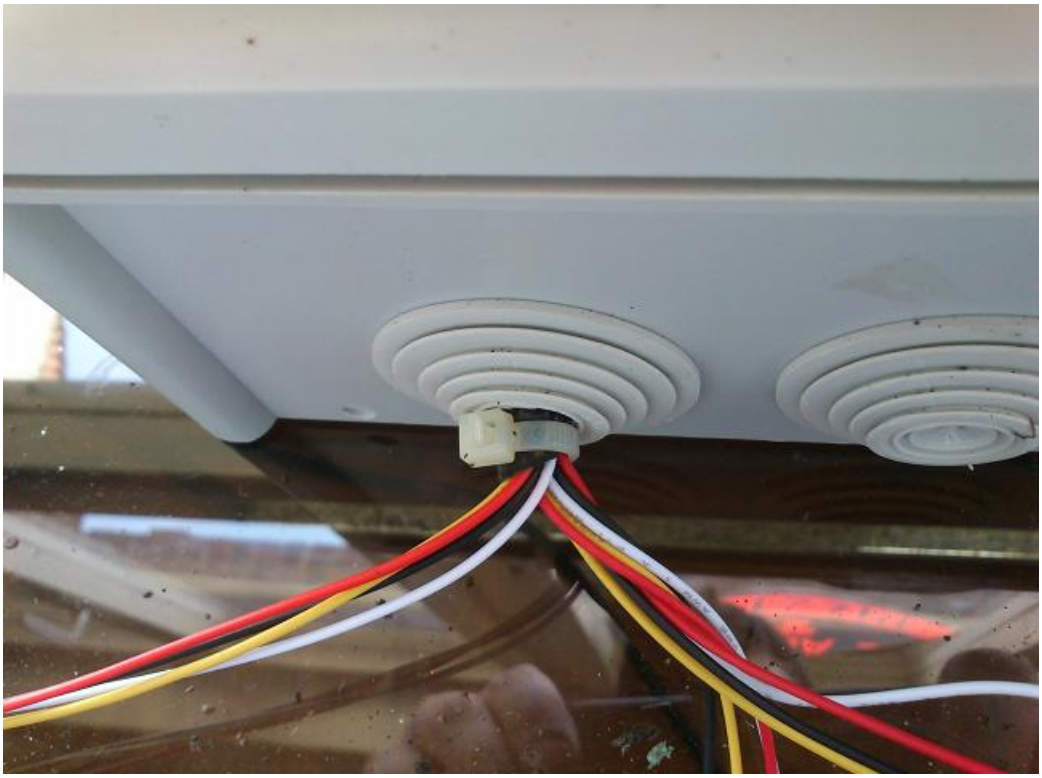


Figura 96 Salida de cables para sensores

Figura 97 Caja estanca redonda con Sensor humedad suelo

Figura 98 Instalación conectada a invernadero de pruebas

CAPÍTULO 8. Presupuesto

1. Hardware y software necesario

Componente	Cantidad	Precio unitario(€)	Precio unitario con 18% IVA (€)	Precio total (€)
Ordenador	1	532.54	628.40	628.40
LabVIEW 2011 licencia campus	-	-	-	Gratuito
Arduino 1.0	-	-	-	Gratuito
Arduino UNO con ATMega328	1	21.90	25.84	25.84
Arduino Shield Conectar y Listo	1	15.75	18.59	18.59
Cables 4 Conductores Shield Conectar y Listo Verde 50cm. 5 ud	2	6.95	8.20	16.40
Módulo Relé 5V Conectar y Listo	4	4.35	5.13	20.52
Módulo RTC Reloj Tiempo Real DS1307 Conectar y Listo	1	9.25	10.92	10.92
Módulo Sensor de Luz Conectar y Listo Analógico	1	4.25	5.02	5.02

Módulo Sensor de Temperatura Conectar y Listo Analógico	1	4.25	5.02	5.02
Módulo Sensor Humedad del Suelo Conectar y Listo	1	7.25	8.56	8.56
Pila CR1225	1	2.08	2.45	2.45
Portafusibles	2	0.36	0.43	0.86
Fusible 3A	1	0.17	0.2	0.2
Fusible 2ª	1	0.17	0.2	0.2
Interruptor	1	0.53	0.63	0.63
Conmutador	1	0.70	0.83	0.83
Regleta	1	0.45	0.53	0.53
Cable rojo y negro	5m	2.14	2.53	2.53
Caja empalmes superficie estanca	1	4.40	5.20	5.20
Caja empalmes redonda 70x36 mm	1	0.68	0.8	0.8
Cable USB 2.0 terminales A y B 1.8m	1	2.08	2.45	2.45
Diodo LED verde	1	0.13	0.15	0.15
Resistencia 100Ω	1	0.17	0.2	0.2
Precio Total		640.91		756.30

Tabla 9 Presupuesto Hardwara y Software

2. Tiempo de Ingeniería

PROYECTO		**HORAS EMPLEADAS**
FASE	**ACTIVIDAD**	
Selección dispositivos	Estudio mercado	38
	Elección	20
Total horas Fase I		58
Programación en Arduino	Funciones individuales	35
	Programación completa	18
Total horas Fase II		53
Programación LabVIEW	Funciones necesarias	26
	Programación	82
Total horas Fase III		108
Construcción física	Estudio inicial	17
	Construcción	10
Total horas Fase IV		27
Informe final	Redacción	75
Total horas Fase V		75
HORAS TOTALES		321

Tabla 10 Presupuesto Ingeniería

La persona que ha diseñado el presente proyecto es un Ingeniero Industrial. El tiempo invertido en la realización del mismo ha sido aproximadamente 321 horas.
El coste horario incluyendo todo tipo de conceptos, tales como seguridad social, vacaciones, impuestos, etc... es de 36 €/hora.

Así pues el Coste de Ingeniería es de: **11556 €**

3. Coste total

Coste	Precio IVA 18% (€)
Hardware y software	756.30
Ingeniería	11556
TOTAL	12312.30€

Tabla 11 Coste total

El coste total del proyecto, costes indirectos no incluidos, asciende a la cantidad de:

DOCEMIL TRESCIENTOS DOCE EUROS CON TREINTA CÉNTIMOS.

CAPÍTULO 9. Conclusiones

1. Conclusiones

En el presente trabajo se ha diseñado e implementado un sistema de control y monitorización para invernaderos domésticos de muy bajo coste (127.9€ es el coste del material, descontado ordenador) y que cumple los objetivos propuestos.

Los sensores y la placa que se han utilizado han sido seleccionados siguiendo los siguientes criterios: disponibilidad del producto, coste y comportamiento.

Las características del sistema son las siguientes:

- Placa Arduino UNO rev3.
- Arduino Shield Conectar y Listo
- Medida de temperatura: El modulo elegido es Módulo Sensor de Temperatura Conectar y Listo Analógico que usa un termistor NTC TTC3A103_34D. Su rango de operación es de -40 a 125ºC con una precisión de +/-1.5%.
- Medida de humedad del suelo: El modulo elegido es Módulo Sensor Humedad del Suelo Conectar y Listo, que es del tipo medida de resistencia del suelo.
- Medida de la luminosidad: El Módulo Sensor de Luz Conectar y Listo Analógico usa una fotorresistencia GL5528 y un amplificador operacional LM358.

Después de diseñar, implementar y verificar el sistema podemos extraer las siguientes conclusiones:

- El diseño del sistema es sencillo sin que requiera apenas acondicionamiento de los sensores, hecho que permite reducir su coste final.

- Se ha diseñado un sistema capaz de conseguir medidas de los parámetros temperatura, humedad del suelo y luminosidad a partir de sensores, crear un sistema de medición en tiempo real de los parámetros anteriormente mencionados, controlarlos de forma automatizada y manual, visualizar el muestreo de los datos por medio de una interfaz de usuario con gráficos en tiempo real y de almacenar los datos en un fichero en Microsoft Excel con su fecha y hora.

2. Líneas de futuro trabajo

En este punto se tratan posibles líneas de trabajo para la ampliación y mejora de este sistema. Las futuras líneas de trabajo son amplias, dependiendo de las necesidades del usuario del sistema.

- Ampliar el número y tipo de sensores que alimentan la placa arduino. Como por ejemplo utilizando sensores de humedad ambiente, sensores de nivel de líquido para un depósito, o más puntos de control para mayores instalaciones.
- Utilización de sistemas de comunicación alternativos entre la placa y el ordenador, como pueden ser conexiones wifi o wireless por medio de diferentes shields para arduino.
- Almacenamiento de datos sin conexión a pc, por medio del shield arduino SD.
- Visualización de los datos sin conexión a pc por medio de pantallas TFT o LCD especialmente diseñadas para arduino.
- Conectar el sistema a internet para el control y monitorización de forma remota.

CAPÍTULO 10. Bibliografía

[1]. J.del Río Fernández, S. Shariat-Panahi, D. Sarrià Gandul, A.M.Làzaro
"LabVIEW Programación para Sistemas de Instrumentación"

[2]. José Rafael Lajara, José Pelegrí
"LabVIEW : entorno gráfico de programación"

[3]. De Silva, Clarence W.
"Sensors and actuators : control systems instrumentation"

[4]. http://en.wikipedia.org/wiki/Resistance_temperature_detector

[5]. http://en.wikipedia.org/wiki/Thermistor

[6]. http://en.wikipedia.org/wiki/Photoresistor

[7]. http://arduino.cc/en/Reference/HomePage

[8]. https://decibel.ni.com/content/groups/labview-interface-for-arduino

[9]. http://zone.ni.com/reference/en-XX/help/371361H-01/

[10]. http://www.ni.com/pdf/manuals/320999e.pdf