Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
Plataformas Abiertas
I ciclo 2019

.

Espacio de crecimiento con control de temperatura y humedad.

Daniela Oporta Abarca
Rafael Angel Murillo Vega
Profesor: Javier Pacheco Brito -

11/Julio/19

Índice

Índice de figuras

1. Objetivos

1.1. Objetivo General

Construir una configuración de elementos eléctricos, controlados mediante una tabla de arduino para automatizar un proceso de riego.

1.2. Objetivos Específicos

Generar una función de codigo que controle la activación de las bombas, y otra que capture las lecturas de los sensores y las almacene.

Crear un código en lenguaje c++ que permita definir los rangos de temperatura y humedad aceptable para la planta.

Plantear una forma recomendada de combinación de los elementos (Sensores y bombas).

2. Funcionalidad

Analizando el hecho de que muchísimas personas que gustan tener sus propios cultivos pero al ser aledañas a las zonas urbanas poseen complicaciones para tener su propio huerto; Ya sea por falta de espacio, tiempo o poco presupuesto para contratar alguien que se encargue de esto. Se plantea un sistema controlado que sea una solución factible a un presupuesto accesible, facilitando tareas mediante el uso de herramientos en desarrollo de software y tecnologías multiplataformas de arduino con uso de lenguaje programable.

3. Introducción

Debido a la influencia de diversos factores, ya sea falta de tiempo, ausencia de espacios óptimos, o en general la falta de disponibilidad que tienen las personas para mantener un huerto en interiores, donde, las condiciones ambientales son determinantes para el crecimiento y desarrollo de estas se propone una solución óptima. Para alcanzar dicho cometido es necesario satisfacer las necesidades del entorno respectivas, el espacio de crecimiento bajo control que se plantea pretende lograr esto mediante el diseño de un sistema de riego automático por medio de una tabla arduino, microcontroladores y lecturas periódicas de la humedad del sustrato en que se encuentra la planta.

Se expone controlar, también con arduino una bomba de agua que regará la planta según un horario establecido, de igual forma de ser necesario se podrán realizar riegos extra cuando lo ameriten las condiciones, y adicionalmente se colocarán dos bombas conectadas a contenedores con nutrientes para las plantas, se propone realizar esto combinando soluciones de hardware y software libres.

Algunos de los elementos se comportan como una señal analógica por su carácter continuo, otros como señal binaria. Según la naturaleza de la señal se deben capturar de una forma diferente.

4. Nota teórica

Primeramente se digitó la palabra reservada o directiva de procesador include, utilizada en lenguaje C para incluir declaraciones de otro fichero o archivos.h así se incluyen éstas en la compilación para organizar el código de manera ordenada, el procesador encuentra la línea include "archivo.h" y reemplaza dicha línea por el código que posee ese archivo o fichero.
Para llevar a cabo funcionalidades que dispone el microprocesador Arduino se agregaron los ficheros Wire.h, Adafruit_Sensor.h, DHT.h de Arduino. La biblioteca DHT.h permite comunicarse con dispositivos I2C / TWI. En las placas Arduino con el diseño R3 (1.0 pinout), SDA (línea de datos) y SCL (línea de reloj) están en los encabezados de pines cerca del pin AREF. El fichero Adafruit_Sensor.h utiliza a los pequeños sistemas integrados al Arduino para recopilar datos de sensores, analizarlos y realizar una acción apropiada o enviar esos datos a otro sistema para procesarlos. Por ultimo el archivo DHT.h nos permite usar un sensor de humedad y temperatura muy fácilmente ya que proporciona una salida de datos digital, sólo lee números enteros.

Se utilizó define, palabra reservada para la definición de constantes en C, en programación, una constante es un valor que no será modificado ni alterado durante la ejecución de un programa, únicamente puede ser leído y utilizado en el código, corresponde a una longitud fija de un área reservada en la memoria principal del ordenador, donde el programa almacena valores fijos. La palabra DHTPIN seguida de un número entero es la que se encarga de definir el PIN del microcontrolador donde se conectará el sensor DHT, mientras que, DHTTYPE se usa para especificar el sensor que está siendo utilizado en el proyecto.

La función void setup esta es la primera en ejecutarse dentro de un programa en Arduino. Es, básicamente, donde se escriben las funciones que llevará a cabo el microcontrolador, escritas en el cuerpo de la función entre las llaves o corchetes. Seguidamente la función void loop que se utilizará como un bucle o ciclo que se repite infinitamente hasta que el microcontrolador sea apagado o reiniciado, sujeto a las condiciones y estructuración de código de acuerdo a lo que se necesita para el programa.

En el setup se establece el comando Serial.begin para indicarle al programa que se iniciará comunicación serial, también llamada secuencial; el proceso de envío de datos de un bit a la vez, de forma cíclica, sobre un canal de comunicación, Serial.begin para lenguaje arduino establece los baudios o la velocidad de datos en bits por segundo para la transmisión de datos en serie. Para comunicarse con el arduino se utilizan las velocidades: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 o 115200.
En el código se hace uso de datos tipo entero para luego utilizarlos en el microcontrolador ya que el DHT utilizado lee números de éste tipo, declarados de forma int _nombre_. Para la impresión en pantalla, en este caso para imprimir en el monitor de serie de arduino se digitó Serial.println el cual muestra en pantalla lo que se escriba entre comillas dentro de los paréntesis que se encuentran concatenados a dicha palabra reservada.

Para que el dispositivo dht sea "enlazado" al código debe establecerse en el monitor de serie la velocidad serial que se utilizará así como en el código mediante dht.begin() o con Serial.begin() estableciendo el número elegido dentro de los paréntesis. Luego para pinMode(_,_) nos piden dos parámetros, el primero, el número de entrada de pin que se utilizará en número entero, luego el modo con el que se comportará el PIN digital. OUTPUT es uno de estos modos, dicho vocablo se utiliza para definir que están en estado de baja impedancia. Esto significa que pueden proporcionar una cantidad sustancial de corriente a otros circuitos.

Se utilizaron punteros de igual forma, variables que contiene la dirección de memoria de datos o de otras variables, inclusive otros punteros. Es decir este tipo de variable apunta al espacio físico donde está el dato o la variable que lo contiene. Un puntero puede apuntar a un objeto de cualquier tipo, en este caso es de tipo entero $_{i}nt * nombre_{.}$

Se establecieron condicionales para manejar el flujo de programa, if { } else {}, en los cuales se escribe una condición dentro de los paréntesis seguidos de if, si está se cumple se ejecuta el código que se encuentra dentro de los corchetes, si ésta no se cumple entonces revisa la condición indicada dentro de los paréntesis de else revisando de la misma forma que, si se cumple se cumpa lo que se encuentra entre corchetes, si no se cumple el programa no ejecuta los cuerpos de las condiciones. Luego, digitalWrite se utiliza cuando un PIN del Arduino ha sido configurado como OUTPUT con pinMode(). se digita dentro de la función el voltaje que de desea manejar en el pin, ya sea HIGH, para valores de 5V o 3,3V o LOW para "suelo" utilizado para apagar el dispositivo, la sintáxis es de la forma digitalWrite(pin,valor) donde los parámetros serán el número de pin y luego para valor HIGH o LOW, no retorna nada.
La función delay pausa el programa por una cantidad de tiempo establecida en milisegundos especificados en la

entrada por parámetro, su sintáxis es delay(ms), y su parámetro ms debe ser un número de tipo entero, no retorna nada.
AnalogRead lee los valores de un PIN análogo específico, la placa de Arduino contiene un multicanal de 10-bit analógo para convertir digitalmente datos, esto quiere decir que escanea entradas de voltajes entre 0 y el voltaje con el que se está operando, ya sea 5V o 3,3V a valores enteros entre 0 y 1023, para Arduino UNO esto se lee mediante una tabla de valores ya asignada, se ingresa por parámetro el nombre o número del PIN que se utilizará.

BOARD	OPERATING VOLTAGE	USABLE PINS	MAX RESOLUTION
Uno	5 Volts	A0 to A5	10 bits
Mini, Nano	5 Volts	A0 to A7	10 bits
Mega, Mega2560, MegaADK	5 Volts	A0 to A14	10 bits
Micro	5 Volts	A0 to A11*	10 bits
Leonardo	5 Volts	A0 to A11*	10 bits
Zero	3.3 Volts	A0 to A5	12 bits**
Due	3.3 Volts	A0 to A11	12 bits**
MKR Family boards	3.3 Volts	A0 to A6	12 bits**

Figura 1: Tabla para asignar pines y voltajes a distintos modelos de Arduino

La función matemática map convierte rangos de números, retorna el número convertido y, su sintaxis para producir porcentajes es de la forma $_(valorescaneado, rangomayor, rangomenor, porcentajemnimo, porcentajemximo)_;$

5. Desarrollo

Enumerando qué funciones se necesitan programar para cubrir las necesidades biológicas básicas de las plantas se programaron ficheros para cada una de estas. Para abastecer a la planta del agua que necesita para vivir en un ambiente adecuado, así como dos sustratos que la nutrirán una vez al día, diariamente se declararon las funciones, variables constantes y se incluyeron las librerías o ficheros de Arduino necesarias para correr el programa de forma correcta.
Las constantes declaradas mediante define definen el número de PIN que se está utilizando en el caso de las DHTPIN, DHTPIN2 y DHTPIN3, la constante DHTYPE es una especificación del tipo de sensor de humedad que se está utilizando en este caso DHT11, las constantes para números enteros declaradas serán utilizadas para establecer márgenes de tiempo, SEGDIA contiene los milisegundos que contiene un día, SEGHIGH los milisegundos contenidos en 10 segundos, SEGSUS los milisegundos contenidos en 5 segundos, para manejar condicionales tipo bool se estableció SI que equivale a un 1 y NO equivalente a 0.
Mediante la palabra reservada DHT se declaran con los nombres dht, dht2 y dht3 las bombas a ser utilizadas, entre sus paréntesis se definen el número de PIN, así como el tipo que éstas utilizan.

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <stdio.h>
#include <stdbool.h>
#include <Arduino.h>
```

Figura 2: Bibliotecas incluidas para el funcionamiento del programa

Seguidamente se declararon las funciones a ser programadas en el archivo fun.cpp lenguaje C++, llamdas void setup(), void loop(), void condicion() y void imprimirLectura().

```
void setup();
void loop();
void condicion(int *puntero);
void imprimirLectura(int *puntero);
```

Figura 3: Declaración de funciones en el archivo fun.h

En el archivo en lenguaje C++ llamado fun.cpp se incluyó el archivo fun.h inicialmente, donde se encuentran las declaraciones de las variables, constantes, y funciones, seguidamente se inició con la programación de la función sin retorno llamada setup, esta no recibe ningún parámetro y en el cuerpo se digitó mediante Serial.begin(115200) que los dispositivos utilizados tipo DHT utilizan comunicación serial con velocidad 115200, luego por medio de pinMode(pin,modo) el número de pin(primer parámetro) y el modo de comportamiento de cada bomba (segundo parámetro).

```
void setup()
{
  //BOMBA1
  dht.begin();
  dht2.begin();
  dht3.begin();
  Serial.begin(115200);
  pinMode(DHTPIN,OUTPUT);

  //BOMBA2
  pinMode(DHTPIN2,OUTPUT);

  //BOMBA3
  pinMode(DHTPIN3,OUTPUT);

  //HIGROMETRO
  Serial.begin(115200);
}
```

Figura 4: Funcion void setup() en fun.cpp

La función tipo void llamada imprimirLectura recibe por parámetro un puntero tipo entero, se declaró dentro de ella un entero llamado porcentaje el cual utilizada el valor numérico contenido en puntero para pasar este a un porcentaje de 1 a 100 en base a valores de 0 a 1023 respectivamente, esta imprime la lectura de valores de humedad para valores de 0 a 1023 mediante el puntero y luego su porcentaje por medio del entero porcentaje donde se ejecuta la función map(par1,par2,par3,par4).

```
void imprimirLectura(int *puntero)
{

   int porcentaje = map(*puntero, 1023, 0,0, 100);
   Serial.print("La lectura de humedad es: ");
   Serial.print(*puntero);
   Serial.print(" de 1023 segun los valores de humedad de la tablatura para dht Arduino");
   Serial.println(" ");

   Serial.print("El porcentaje de la humedad es del: ");
   Serial.print(porcentaje);
   Serial.print.("%");
   Serial.println(" ");
}
```

Figura 5: Código para impresión de humedad mediante porcentajes (Uso de función map) y valores de 0 a 1023

La función condicion recibe por parámetro un puntero tipo entero, esta no retorna nada, dentro de la misma se encuentra un entero llamado condicion donde se guardaran valores para un condicional booleano, se declaró un puntero tipo entero llamado *ptr este apunta al espacio de memoria donde se encuentra condicion, se usa esta lógica para mediante el puntero asignar luego valores a la condicion.
Por medio de un condicional if else y, con el valor numérico al que apunta puntero, es decir los valores de humedad de 0 a 1023 de acuerdo a la tabla de Arduino se dicta que, de acuerdo a este número se cree una comparación dentro de los ifs para determinar si el sensor escanea un suelo está seco, húmedo, sumergido en agua o fuera de la tierra, de acuerdo a esta condición se determina si la bomba de agua debe activarse o no, si debe activarse esto sucede durante solamente 10 segundos también lo lógica dicta si en el entero condicional mediante el puntero asignado se guarde un SI o NO. El uso de la variable condición se utiliza en un if else que determina que, si esta es igual a SI se activen las bombas 2 y 3 que abastecerán de sustratos a la planta una vez al día durante 5 segundos, en caso de que la variable sea NO las bombas de sustrato serán apagadas. Sus sintaxis son de tipo digitalWrite (PIN, modo) donde PIN será el número entero definido en fun.h para el pin donde se está trabajando y modo podrá contener HIGH en el caso de encender las bombas o LOW parar apagar estas.

```
void condicion(int *puntero){

   int condicion;
   int *ptr;
   ptr=&condicion;


 if (*puntero>= 1000) {
  Serial.println("El sensor esta desconectado o fuera del suelo");
  Serial.println("Se desactivara el sistema de riego");
  digitalWrite(DHTPIN, LOW);
  *ptr=NO;
}
  else if (*puntero<1000 && *puntero>=600)
 {
  Serial.println("El suelo esta seco");
  Serial.println("Se iniciara el sistema de riego");
  digitalWrite(DHTPIN, HIGH);

  delay(SEGHIGH);

  digitalWrite(9, LOW);
  *ptr=SI;
}
```

Figura 6: Primera parte funcion condición

```
else if(*puntero <600 && *puntero >= 370)
{
 Serial.println("El suelo esta humedo");
 Serial.println("El sistema de riego se desactivara");
 digitalWrite(DHTPIN, LOW);

 *ptr=SI;
 }
else if (*puntero <370)
{
 Serial.println("El sensor esta sumergido en agua");
 Serial.println("El sistema de riego se desactivara");
 digitalWrite(DHTPIN, LOW);
 *ptr=NO;
 }

 if(*ptr=SI)
 {
   digitalWrite(DHTPIN2, HIGH);
   digitalWrite(DHTPIN3, HIGH);
   delay(SEGSUS);
   digitalWrite(DHTPIN2, LOW);
   digitalWrite(DHTPIN3, LOW);
   Serial.println("Se activara el riego diario de los dos sustratos");
   }
 else if(condicion=NO){
```

Figura 7: Segunda parte funcion condición

```
 else if(condicion=NO){
   digitalWrite(DHTPIN2, LOW);
   digitalWrite(DHTPIN3, LOW);
    Serial.println("No se activara el riego diario de los dos sustratos");
   }


 //el sistema de riego se ejecuta una vez al dia

}
```

Figura 8: Tercera parte funcion condición

La función loop() no tiene retorno, no recibe nada por parámetro actúa como un buble o ciclo, este es el main del programa, en ella se declaró un entero donde se inicializa el valor de lectura que recolecta el sensor mediante la función analogRead(A0) la cual tiene por parámetro A0, es decir el lugar de la placa Arduino donde está operando. Se declaro luego un puntero tipo entero el cual apunta a la dirección de memoria de la variable lectura, este puntero es utilizado como parámetro en el loop para las funciones que reciben por parámetro esto: imprimiLectura y condición, finalmente dentro del loor se establece una pausa por medio de la función delay de los milisegundos que posee 24 horas para que el ciclo corra de forma diaria.

```
void loop()
{
  int lectura = analogRead(A0);
  int *puntero=&lectura;

  imprimirLectura(puntero);
  condicion(puntero);

  delay(SEGDIA);
  }
```

Figura 9: (Funcion Loop en fun.cpp)

6. Conclusiones

La implementación de un ambiente controlado o un pequeño sistema de riego es de ayuda en términos de optimización del tiempo y la posibilidad de realizar el riego de un huerto doméstico a un bajo costo. Mediante pruebas a este se puede comprobar su utilidad en tiempo real para lapsos en milisegundos equivalentes a 24 horas debido a sus intervalos en envío y recepción de información programados para ocurrir únicamente una vez al día. El funcionamiento de los sensores de humedad presenta una precisión aceptable, su intercambio de datos es de gran certeza. Mediante la digitalización de los valores de humedad se logró establecer un condicional de acuerdo con el riego que la planta necesita. Los resultados obtenidos validan el supuesto planteado en los objetivos donde se dicta que, con la implementación de un sistema autónomo para el riego de plantas, se minimiza el trabajo de las personas y se obtendrá un eficiente uso de agua. A partir de los efectos obtenidos se denota cómo la implementación de este sistema permitirá un uso más consciente del agua basado en el balance de la humedad del suelo y temperatura del aire que afecta este, y realizando el riego de sólo ser necesario, esto implica una reducción significativa del consumo de agua. El proyecto está delimitado a espacios reducidos, por su ejecución para un bajo consumo de potencia, pero demuestra como mediante la tecnología se pueden solucionar problemas de tiempo para optimizar este y mejorar el estilo de vida de las personas.

Referencias

[1] Escalas Rodríguez, G. (2015). Diseño y desarrollo de un prototipo de riego automático controlado con Raspberry Pi y Arduino (Bachelor's thesis, Universitat Politècnica de Catalunya), pp. 12-45.

[2] Cuenca Cahuana, G. M. (2016). Implementación de un Sistema de Riego Automático y Manual para Optimización de recursos con Adquisición de datos de Sensor de Humedad en Computador, pp. 1-3.

[3] Alfonso A. GUIJARRO-Rodríguez. (2018, Mayo 19). Sistema de riego automatizado con arduino. Revista Espacios, 39, 1-15. 2019, Mayo 29, De Espacios Base de datos.

[4] Evans, Brian W., (2007) Arduino Programming NOtebbok

[5] Santo Oncero David (2007) <Hardware Libre>, Todo Linux, Madrid: Studio Pres. Pp 21-12

[6] Arduino Homepage, disponible en: http://www.arduino.cc.es/