INSTITUTO TECNOLÓGICO DE COSTA RICA

ESCUELA DE INGENIERÍA ELECTRÓNICA

ARQUITECTURA DE COMPUTADORAS

SEGUNDO SEMESTRE DEL 2018

Proyecto 3 Desarrollo de un invernadero utilizando sistemas empotrados

Profesor: Ronny García Ramírez

Estudiantes:

Jordan Cantillano Quirós

Jacobo De Bruyn Monge

Felipe Solano Sanchez

Grupo: 1

Sede Central, Cartago

Índice

Descripción del diseño de software

Receptor

El algoritmo principal del receptor puede definirse como el siguiente, en el cual se inicia un contador, se lee la información proveniente de la pc, si es que existe alguna , hasta que este contador sea igual o mayor a los 5 ms, una vez pasa esto se lee la información proveniente del módulo emisor, si es que existe alguna, se realiza una comparación en donde si el valor recibido de la pc, el cual es el valor límite es menor que el valor emitido por los sensores, entonces el circuito actúa sobre el sistema.
Y finalmente imprime en pantalla la información tanto de los sensores como la de los valores límites.

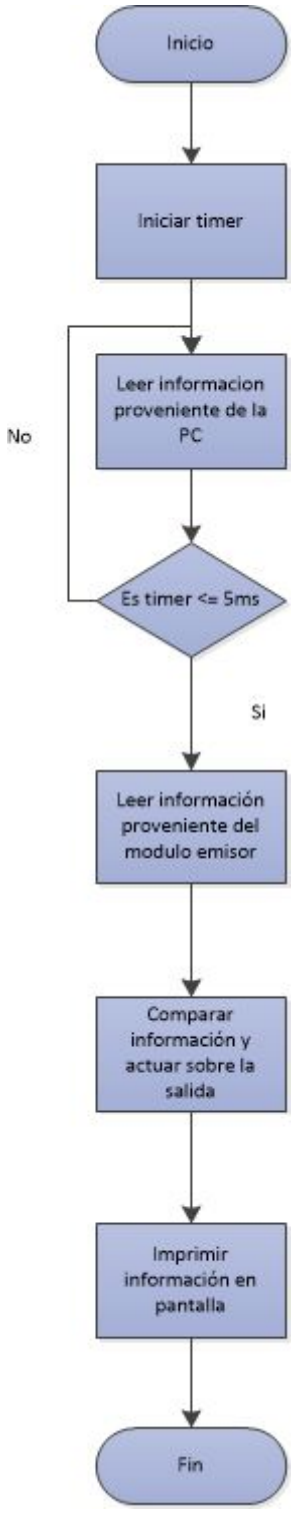

Figura 1. Diagrama general del receptor

1. Leer información proveniente de la PC

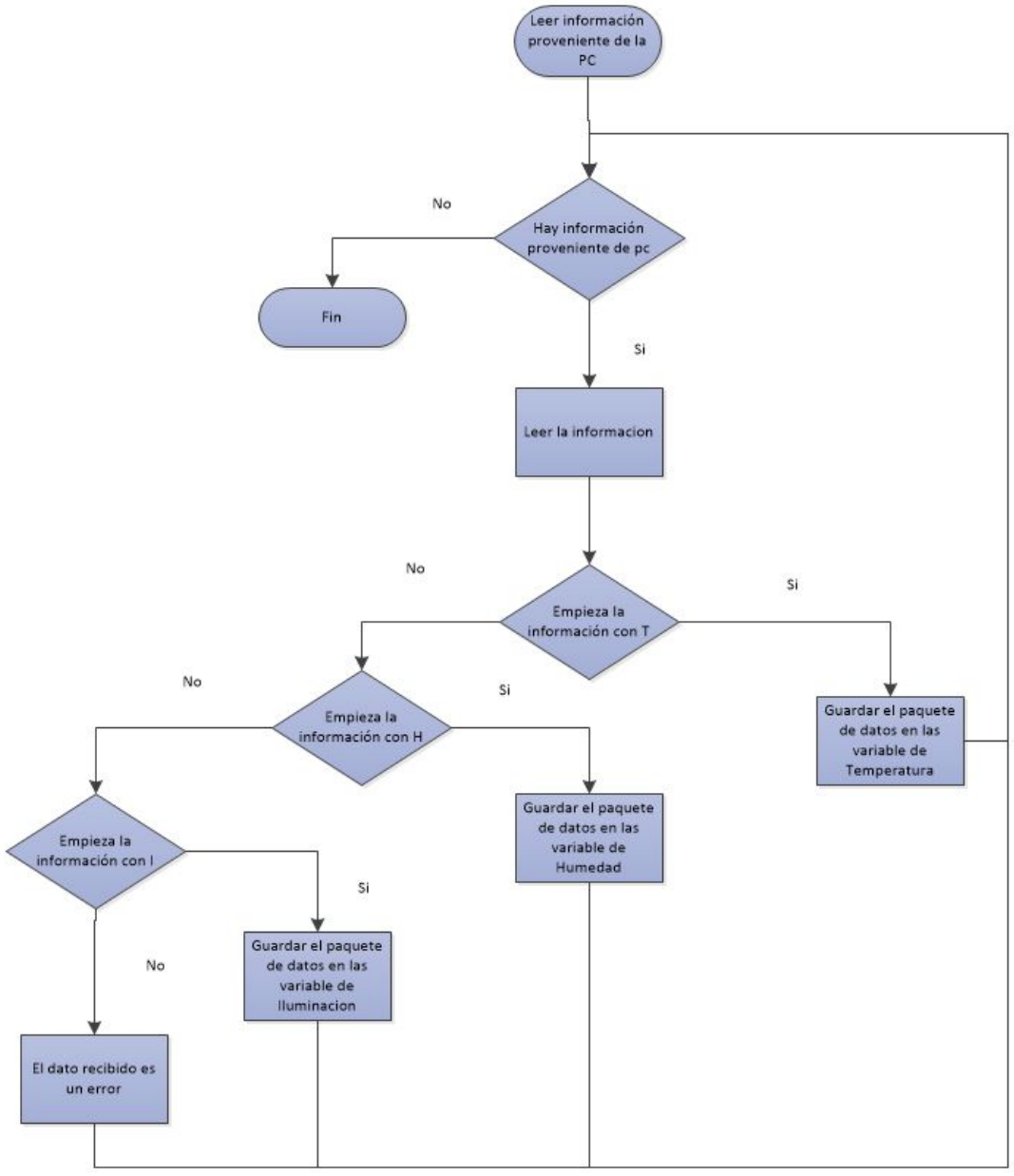

Figura 2. Diagrama del algoritmo de lectura de datos del PC

2. Leer información proveniente del módulo emisor

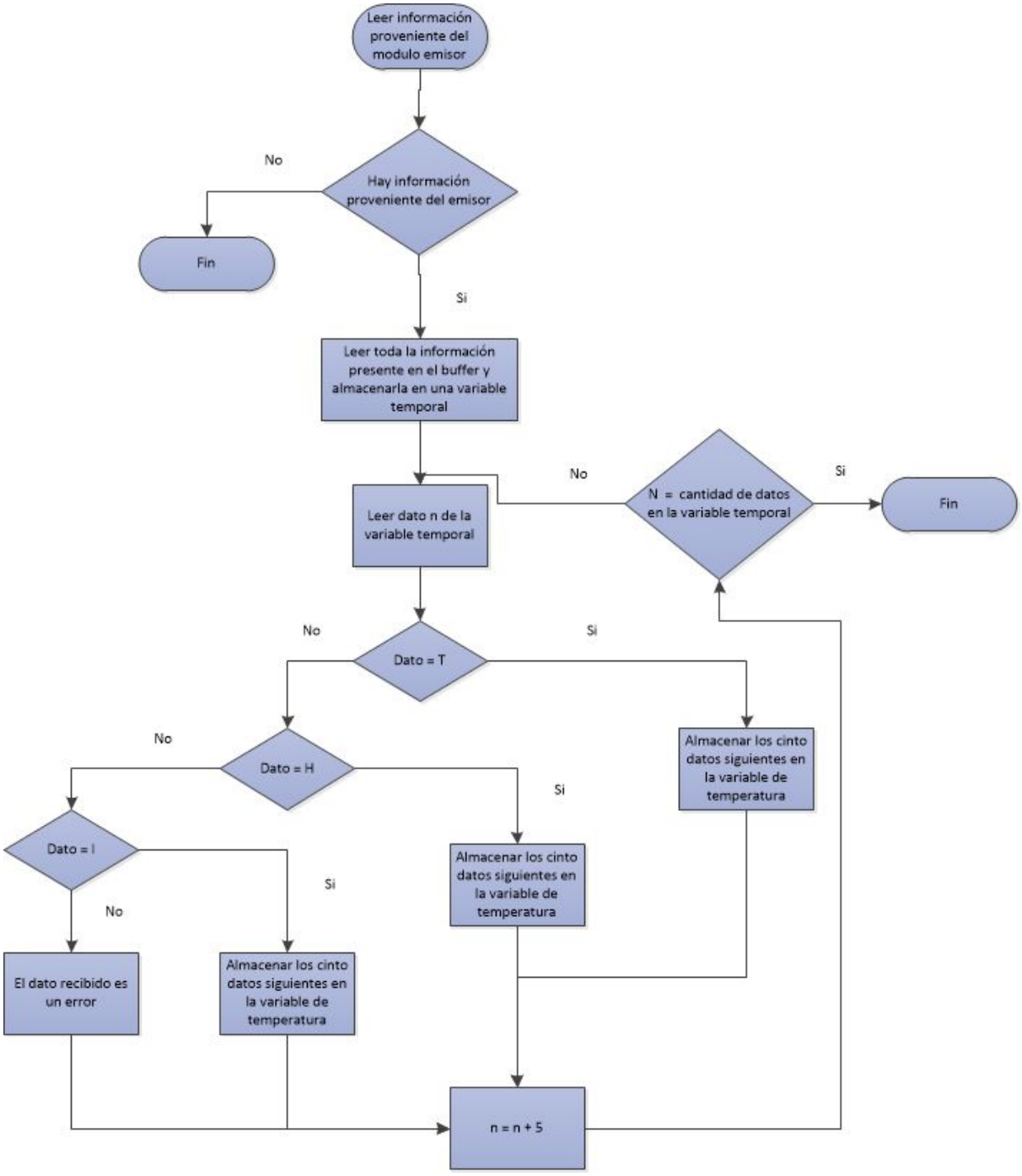

Figura 3. Diagrama del algoritmo de lectura del módulo inalámbrico

3. Comparar la información y actuar sobre las salidas

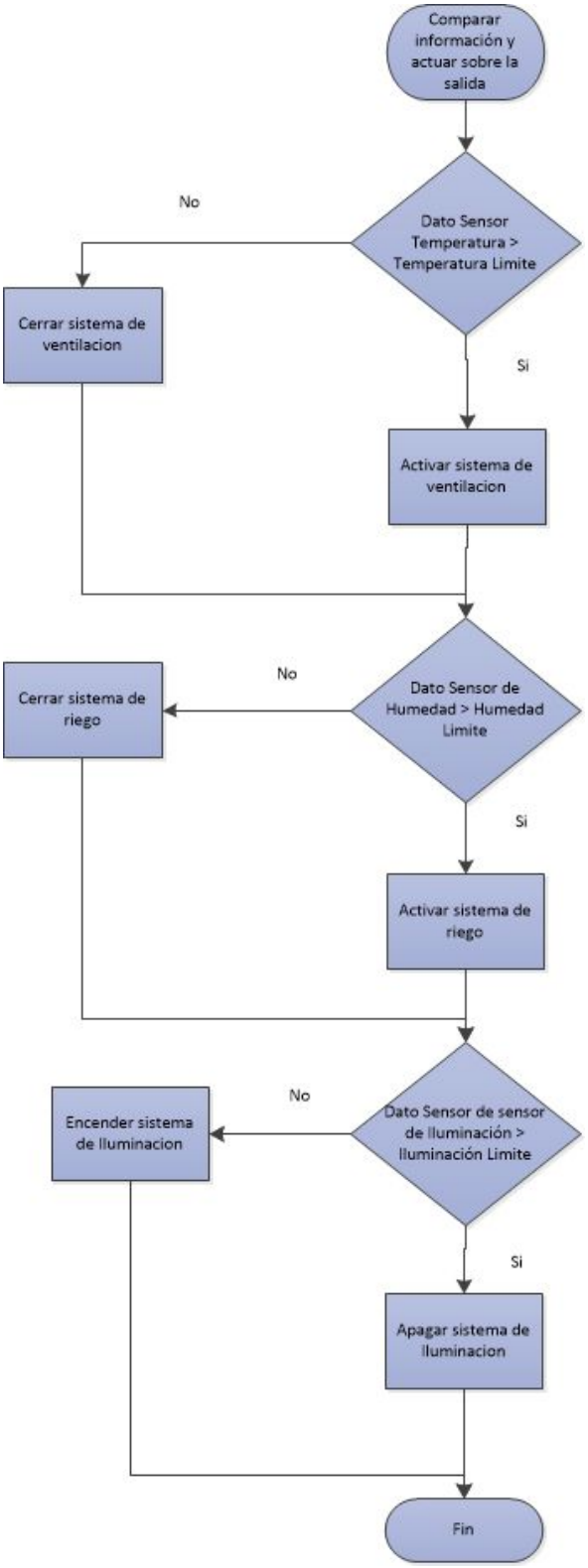

Figura 4. Diagrama del algoritmo de control para los actuadores

Sensores

Con respecto a las entradas del sistema, estas son tres y representan mediciones de temperatura, luminosidad y humedad. Se utiliza un fotoresistor GL5516 para realizar las mediciones de luminosidad y para las mediciones de temperatura y humedad se simulan valores de voltaje entregados por un sensor para cada variable mediante potenciómetros.
En la Figura - se muestran los componentes utilizados y sus conexiones con el módulo transmisor.

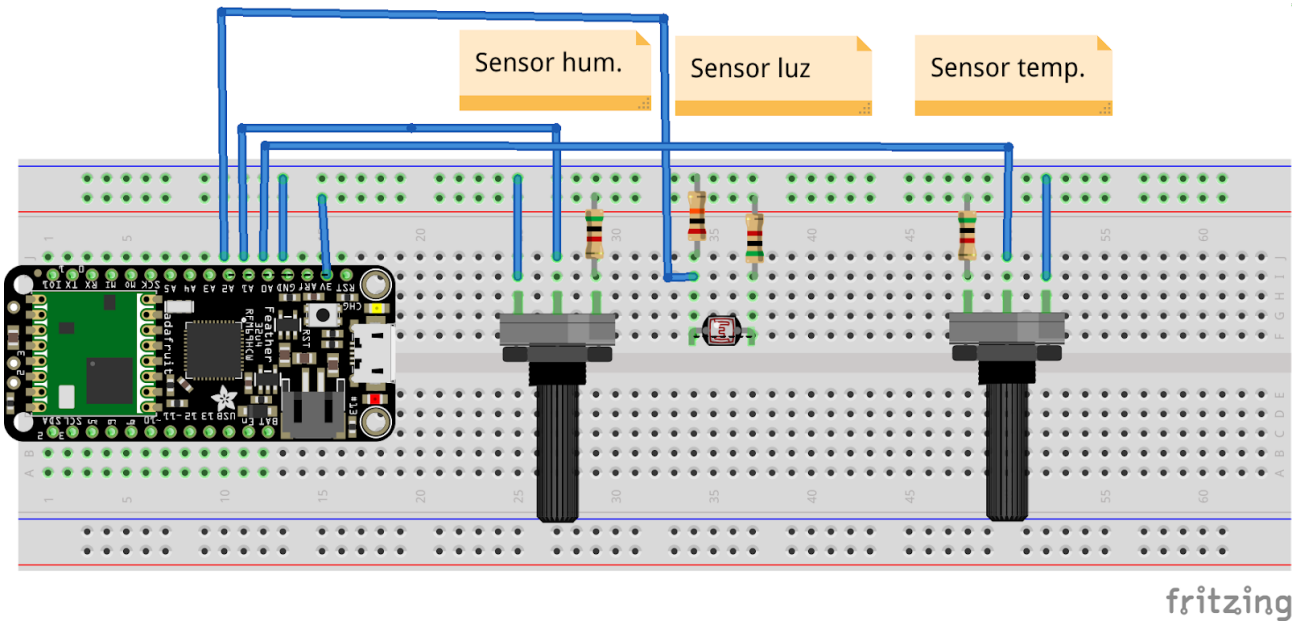

Figura 5. Esquema de los sensores implementado en el software Fritzing.

Puede observarse que para el caso de los sensores simulados se realiza un divisor de tensión con una resistencia de 5 kΩ y un potenciómetro de 10 kΩ. Se alimenta entonces la rama del divisor de tensión con 3.3 V proporcionados por uno de los pines del Arduino. Con esto se logra que al sensar el voltaje en la patilla del medio del potenciómetro se tenga un voltaje de 0 a 2 V aproximadamente, lo cual es uno de los requerimientos del sistema propuesto.
En cuanto a la medición por medio del sensor de luz, se consulta la hoja de datos del fabricante y se encuentra la relación de que el sensor varía de 5-10 kΩ en presencia de 10 lux (valor cercano al esperado en ambientes cerrados) a 0.5 MΩ en la oscuridad. Con esto se diseña el sistema para que en condiciones de oscuridad el fotoresistor tome prácticamente todo el voltaje dado por la alimentación. Con respecto a que se aumenta el nivel de iluminación esta resistencia disminuye, llegando al límite de 5 kΩ aproximadamente Este dato se toma experimentalmente, ya que como se mencionó anteriormente en la hoja de datos el fabricante da un rango dentro del cual puede estar el valor mínimo de resistencia. Al llegar a este valor quedan en la rama un equivalente de dos resistencias: una de 10 kΩ formada por una de 5kΩ en serie con el fotoresistor (el cual tiene el valor de 5kΩ)

y otra resistencia de 20 kΩ, en la cual se presenta un tercio del voltaje de alimentación, aproximadamente 2 V, cumpliendo entonces con lo requerido para el sistema de sensado.
Posteriormente en el Arduino se toman estos valores analógicos y se les aplican una serie de transformaciones para llegar a una escala y precisión deseadas. En primer lugar debido a que se utiliza un ADC para la conversión de las entradas analógicas, se obtiene en el Arduino un valor en una escala de 0 a 1023, esto debido a que el ADC interno de este cuenta con 10 bits. Por lo tanto primero se deben mapear los valores digitales obtenidos que van de 0 a 445 aproximadamente a valores de entre 0 a 2 V.
Se utiliza una función creada en el código para esta y la siguiente conversión o mapeo, llamada fmap. Esto debido a que aunque el lenguaje de programación de Arduino cuenta con una función para esto (map), esta sólo trabaja con números enteros y debido a la aplicación del presente sistema es necesario el uso de valores del tipo float.
Posteriormente se realiza otro mapeo hacia una escala que refleje los valores reales de acuerdo a mediciones realizadas previamente de los valores límites con los que se trabaja. Cada uno de los valores medidos se concatena a una letra que representa el origen de la medición.

Transmisor
Para el algoritmo de transmisión de datos se divide en tres partes, la recolección de los datos, la codificación y formato de los datos y la transmisión inalámbrica de estos datos. Se realiza estas tres partes en el orden anterior y se repite el bucle.

- **Recolección e interpretación de datos:** se detalló esta etapa en la sección anterior correspondiente a los sensores.

- **Codificación y formato de datos:** Dependiendo de la fuente de cada dato se le asigna una letra. Si el dato viene del sensor de luz al dato se le concatena la letra I al inicio de este, si viene del sensor de humedad se le concatena la letra H y por último si el dato viene del sensor de temperatura se le concatena la letra T.

  Seguidamente se crea una variable llamada Dato que es de tipo string que sirve para guardar la concatenación de los tres datos de cada sensor con su respectiva letra.

  Por último se crea la variable dato RF la cual es de tipo Arreglo de Char el cual guarda la información de la variable anterior llamada Dato en este formato para estar lista para la transmisión. Se convierte los datos de tipo string a arreglo de char por medio de la función del objeto string llamada toCharArray().

- **Transmisión inalámbrica:** Inicialmente se utiliza la librería RadioHead, después se configura los pines a utilizar y se escoge la configuración correspondiente al dispositivo en uso en nuestro caso es el procesador Feather 32u4, se configura también la frecuencia de transmisión la cual en nuestro caso es de 915 MHz. La configuración anterior se hace solo en al inicio y no es parte del bucle.

Por último se manda la variable datoRF por medio de la función de la librería utilizada con el comando: *"rf69.send((uint8_t *)datoRF, strlen(datoRF))"* y después para asegurarse de darle el tiempo necesario a la transmisión se utiliza el comando: "*rf69.waitPacketSent();"* y de esta manera ya se transmitió los datos.

Principales retos a resolver y su resolución

En primer lugar se contó con el inconveniente de conseguir una precisión para una escala con decimales, esto debido a que en primer lugar se encontraron fuentes en donde se explicaba el comando map, ampliamente utilizado para la conversión de lecturas analógicas a valores de voltaje. Sin embargo, como se explicó anteriormente, esta función redondea a valores enteros, por lo cual no era adecuada para esta aplicación que requiere precisión con decimales.
Luego de investigar se encuentra que un método para resolver este problema, es el uso de una función que linealice los valores dentro de un rango a otro rango, pero definiendo los valores de entrada y salida como float [6]. La ecuación de esta linealización es la siguiente:

$$resultado = \frac{(valor\ original - min_e)(max_s - min_s)}{(max_e - min_e + min_s)}$$

Donde los subíndices representan:

e: valores de entrada, en la escala original
s: valores de salida, en la escala a la cual se pasa el resultado.

Por otra parte uno de los retos a resolver con respecto a la comunicación entre módulos fue el de leer la información proveniente del módulo emisor en el módulo receptor y compararla con la información proveniente la PC. Para solucionar esto se optó por crear un estándar para la compatibilidad de los datos en el cual los datos debían de seguir el siguiente formato:

- Para enviar un dato referente a la temperatura se debe escribir una T (mayúscula) y pegado el dato en Grados Celsius con dos decimales, sin espacio alguno.
  Solo acepta valores con dos decimales
  La parte entera debe ser de dos dígitos, ni más ni menos.
  Ejemplo
  T37.52
  T40.00
  T59.70
  T00.00
  T01.05

- Para enviar un dato referente a la humedad se debe escribir una H(mayúscula) y pegado el dato de la humedad, este debe ser un porcentajes, con dos decimales, sin espacio alguno.
  Solo acepta valores con dos decimales

La parte entera debe ser de dos dígitos, ni más ni menos.
Ejemplo

H37.52	37.52% de humedad
H40.00	40.00% de humedad
H59.70	59.70% de humedad
H01.04	1.04% de humedad

- Para enviar un dato referente a la iluminación se debe escribir una I(mayúscula) y pegado el dato de la Iluminación,este debe ser un porcentajes, con dos decimales, sin espacio alguno.
  Solo acepta valores con dos decimales
  La parte entera debe ser de dos dígitos, ni más ni menos.
  Ejemplo
  I37.52

Finalmente otro reto fue mandar los tres datos mediante la librería RadioHead,ya que se tenía la información de los tres sensores concatenadas en formato de string y para mandar el dato por medio de esta librería tiene que estar en formato de arreglo de caracteres. esto se soluciona muy fácil al utilizar la función de los objetos string de arduino llamada toCharArray() la cual se utiliza de la siguiente manera: string.toCharArray(buf, len).[1]

Propuestas de mejoras y su posible implementación

Se puede visualizar cuando se transmite y se recibe los datos por medio un LED en cada Arduino y se implementa por medio una función en el receptor que responda al transmisor que llegó correctamente el mensaje y parpadee el LED y seguidamente en el transmisor se detecte la respuesta y la función haga parpadear el LED cuando se detecta la respuesta.

Podría modificarse la acción de los actuadores, donde se tiene en vez de un límite para cada sensor tener un rango y así tener dos acciones diferentes para cada medición física por si supera el rango o decrece fuera del rango.

Con respecto a los sensores, podría realizarse un sistema que de acuerdo a datos previamente obtenidos vaya mejorando su resolución en ciertos rangos de medidas. Es decir, que por sí sólo el sistema logre adaptarse a las condiciones a las que es expuesto. Esta mejora podría implementarse mediante algoritmos que analizan los datos de las mediciones cada cierto periodo de tiempo y que estén programados para realizar cambios en parámetros del código de acuerdo a estos, por ejemplo en los valores de mapeo de las mediciones realizadas.

Referencias Bibliográficas

[1] Arduino.cc. (2018). *Arduino Reference*. [online] Available at: https://www.arduino.cc/reference/en/language/variables/data-types/string/functions/tochararray/ [Accessed 9 Nov. 2018].

[2] Learn.adafruit.com. (2018). *Linux Setup | Adafruit Arduino IDE Setup | Adafruit Learning System*. [online] Available at: https://learn.adafruit.com/adafruit-arduino-ide-setup/linux-setup#udev-rules [Accessed 8 Nov. 2018].

[3] Learn.adafruit.com. (2018). *Using the RFM69 Radio | Adafruit Feather 32u4 Radio with RFM69HCW Module | Adafruit Learning System*. [online] Available at: https://learn.adafruit.com/adafruit-feather-32u4-radio-with-rfm69hcw-module/using-the-rfm69-radio [Accessed 9 Nov. 2018].

[4] Learn.adafruit.com. (2018). *Using with Arduino IDE | Adafruit Feather 32u4 Basic Proto | Adafruit Learning System*. [online] Available at: https://learn.adafruit.com/adafruit-feather-32u4-basic-proto/using-with-arduino-ide [Accessed 8 Nov. 2018].

[5] Senba Optical & Electronic Co. *GL55 Series Photoresistor.* [online] Available at: http://files.microjpm.webnode.com/200000821-dc694dd605/GL55%20Series%20Photoresistor.pdf [Accessed 8 Nov. 2018].

[6] Arduino.cc (2018). *map() and fmap() function.* [online] Available at: https://forum.arduino.cc/index.php?topic=107106.0 [Accessed 8 Nov. 2018].