**UNIVERSIDAD NACIONAL DE TRUJILLO**

**FACULTAD DE INGENIERÍA**

**ESCUELA DE INGENIERÍA MECATRÓNICA**

****

**IMPLEMENTACIÓN de UN SCADA INTELIGENTE basado en tecnologías web PARA un sistema doméstico de cultivo AGRÍCOLA de ambiente controlado**

**TESIS**

**PARA OBTAR EL TÍTULO PROFESIONAL DE**

**INGENIERO MECATRÓNICO**

**Autor:**

**Pereyra Gonzáles, Irvin Jair**

**Asesor:**

**Ing. León Lescano, Edward Javier**

**Trujillo - Perú**

**2018**

**Contenido**

[DEDICATORIA 3](#_Toc517797343)

[AGRADECIMIENTO 4](#_Toc517797344)

[RESUMEN 5](#_Toc517797345)

[ABSTRACT 6](#_Toc517797346)

[1. INTRODUCCIÓN 7](#_Toc517797347)

[1.1. Realidad problemática 7](#_Toc517797348)

[1.2. Formulación del problema 7](#_Toc517797349)

[1.3. Justificación del estudio 7](#_Toc517797350)

[1.4. Antecedentes 9](#_Toc517797351)

[1.5. Objetivos 10](#_Toc517797352)

[1.5.1. General 10](#_Toc517797353)

[1.5.2. Específicos 10](#_Toc517797354)

[1.6. Marco teórico 11](#_Toc517797355)

[1.7. Marco conceptual 11](#_Toc517797356)

[2. MARCO METODOLÓGICO 11](#_Toc517797357)

[2.1. Hipótesis 11](#_Toc517797358)

[2.2. Variables 11](#_Toc517797359)

[2.2.1. Variables independientes 11](#_Toc517797360)

[2.2.2. Variables dependientes 11](#_Toc517797361)

[2.3. Metodología 15](#_Toc517797362)

[2.3.1. Tipo de estudio 15](#_Toc517797363)

[2.3.2. Diseño 15](#_Toc517797364)

[2.4. Población y muestra 15](#_Toc517797365)

[2.4.1. Población 15](#_Toc517797366)

[2.4.2. Muestra 15](#_Toc517797367)

[2.4.3. Método de investigación 15](#_Toc517797368)

[2.5. Técnicas e instrumentos de recolección de datos 15](#_Toc517797369)

[2.5.1. Técnicas 15](#_Toc517797370)

[2.5.2. Instrumentos 15](#_Toc517797371)

[2.6. Métodos de análisis de datos 15](#_Toc517797372)

[3. DESARROLLO Y RESULTADOS 15](#_Toc517797373)

[4. DISCUCIÓN 32](#_Toc517797374)

[5. CONCLUSIONES 32](#_Toc517797375)

[6. SUGERENCIAS 32](#_Toc517797376)

[7. BIBLIOGRAFÍA 32](#_Toc517797377)

[8. ANEXOS 32](#_Toc517797378)

# **DEDICATORIA**

# **AGRADECIMIENTO**

# **RESUMEN**

# **ABSTRACT**

# **CAPÍTULO I - INTRODUCCIÓN**

## **Realidad problemática**

## La falta de educación, la desorganización y el uso inadecuado de los recursos naturales por parte de muchos agricultores genera desequilibrios en el ecosistema que afectan a la agricultura sostenible. Así también, la casi nula transferencia tecnológica y la continua aparición de nuevos minifundios limitan la explotación de terrenos; entorpeciendo los procesos logísticos y en general, la dinámica del mercado agrícola.

## La agricultura debe proveer los alimentos en la cantidad y la calidad necesarias para una vida sana; no obstante, el tema de la seguridad alimentaria implica no sólo mayor producción y productividad sino también una clara conciencia en los consumidores sobre como alimentarse mejor. La insuficiencia de alimentos en cantidad y calidad asociado con malos hábitos alimenticios repercuten sobre la calidad de vida del habitante peruano, es por ello que los trabajos en materia de seguridad alimentaria tienen aún mucho por delante. (Problemas en la agricultura peruana, MINAGRI, 15 de abril de 2018, Disponible en: http://minagri.gob.pe/portal/?id=190&start=3).

## Desde un enfoque más global, es necesario darse cuenta que el mundo actual necesita dejar de lado muchos sistemas socioeconómicos que van mostrando signos de obsolescencia. En torno a la agricultura, a parte de los procesos propios de cultivo respecto de algún vegetal; existen muchos más factores a considerar para garantizar el bienestar de los productores primarios y establecer una agricultura sostenible a largo plazo. Por ejemplo, los más beneficiados con la comercialización de los productos de primera necesidad, no son necesariamente los agricultores, y sus ganancias siempre son afectadas por los costos elevados del transporte. Y a pesar de que se planteen distintas reformas que puedan aliviar un poco esta situación, la sobrepoblación y la escasez de recursos naturales en los próximos años, como el agua o el petróleo, terminará por obligar a las sociedades a plantearse soluciones que la tecnología actual ya permite hacer realidad. En tal sentido, puesto que dichas soluciones demuestran ser incluso más eficientes que nuestros sistemas agrícolas tradicionales, para mejorar la situación de la agricultura en nuestro país sería oportuno ir implementando Sistemas de Agricultura de Ambiente Controlable a todo nivel, desde entornos domésticos o urbanos hasta entornos industriales, y hacer más precisos los rangos de sus parámetros ambientales de trabajo.

## **Formulación del problema**

## ¿Cómo determinar mejores condiciones climáticas en sistemas domésticos de cultivo agrícola de ambiente controlado?

## **Justificación del estudio**

### **Relevancia Tecnológica**

## En centros de investigación, desarrollo e innovación alrededor del mundo se están implementando nuevas alternativas como la presentada en este trabajo, que cambian y demuestran resultados prometedores respecto del sistema con el que se ha venido manejando la agricultura tradicional. Por lo tanto, es conveniente aprovechar la gran variedad de opciones tecnológicas disponibles en el mercado para comenzar a explorar los beneficios de su aplicación en nuestra realidad.

### **Relevancia Institucional**

## El presente trabajo tiene la capacidad de atraer equipos de investigación multidisciplinarios. Por ejemplo, que conlleven la participación de interesados en los campos de la física, matemática, ciencias de la computación, ingeniería, gestión, economía, etc.

### **Relevancia Social**

## Con los nuevos métodos agrícolas se pretende también que cada individuo viva participando activamente en un entorno donde las actividades agrícolas sean autosostenibles. Lo que implica, a su vez, una elevación de la cultura alimentaria.

### **Relevancia Económica**

## La agricultura vertical también implica desligar los procesos logísticos y de transporte que actualmente son imprescindibles para que los agricultores puedan vender sus productos, y que actualmente es común que la variación del costo del petróleo influya negativamente en sus ganancias.

### **Relevancia Ambiental**

## Kurt Benke y Bruce Tomkins (2017) afirman que las megatendencias mundiales de la disminución del suministro de agua, el aumento de la población, la urbanización y el constante cambio climático han contribuido a la disminución global de las existencias de tierras cultivables por persona. En estas circunstancias, es probable que la sostenibilidad del modelo agrícola tradicional basado en grandes granjas rurales se vea amenazada en las próximas décadas. Un enfoque para abordar este problema desafiante es la agricultura vertical, que se basa en la agricultura de ambiente controlado y diseños de invernadero adecuados para entornos urbanos (p. 12). La agricultura vertical se ha demostrado a escala piloto y también a nivel de producción y tiene ventajas potenciales sobre la agricultura rural, incluido el uso de la hidroponía, que desafía la necesidad de una agricultura basada en el suelo para una variedad de cultivos. Los beneficios potenciales de la agricultura vertical incluyen un modelo sostenible de producción de alimentos con producción de cultivos durante todo el año, mayores rendimientos en un orden de magnitud y ausencia de sequías, inundaciones y plagas (p. 13).

## **Antecedentes**

## Muhammad Ikhwan y Norashikin M. Thamrin (2018) presentan un proyecto cuyo objetivo principal es construir un sistema para controlar la humedad del suelo y controlar el contenido de agua a través del navegador web en la computadora portátil, el teléfono móvil y otros dispositivos portátiles y compactos (p. 1).

## Deepak Vasisht, Zerina Kapetanovic, Jongho Won, Xinxin Jin, Ranveer Chandra, Ashish Kapoor, Sudipta N. Sinhaand Madhusudhan Sudarshan, Sean Stratman (2017) afirman que las técnicas basadas en datos ayudan a impulsar la productividad agrícola al aumentar los rendimientos, reducir las pérdidas y reducir los costos de los insumos. Sin embargo, estas técnicas han visto una adopción escasa debido a los altos costos de recopilación manual de datos y soluciones de conectividad limitadas (p. 1).

## Yap Shien Chin y Lukman Audah (2017) afirman que la agricultura vertical es difícil de practicar porque los cambios menores en el entorno dejarían un gran impacto en la productividad y la calidad de la actividad agrícola. Por lo que, presentan un estudio con el objetivo de proporcionar un sistema de monitoreo agrícola vertical para ayudar a mantener el seguimiento de las condiciones físicas de los cultivos (p. 1).

## Kurt Benke y Bruce Tomkins (2017) afirman que existe la necesidad de aumentar los fondos para la investigación en genética vegetal para optimizar el rendimiento, ampliando la gama de tipos de cultivos y ajustando para obtener una respuesta óptima a variables controladas como la longitud de onda de la iluminación LED, la temperatura, la humedad y los niveles de CO2 (p. 14).

## Los resultados demuestran claramente que los sistemas de cultivo verticales (VFS) presentan una alternativa atractiva a los sistemas de crecimiento hidropónico horizontal y sugieren que se podrían lograr mayores aumentos en el rendimiento mediante la incorporación de iluminación artificial en el VFS.'' (Dionysios Touliatos, Ian C. Dodd y Martin McAinsh, 2016, p. 1).

## Malek Al-Chalabi (2015) afirma que los hallazgos indican que la agricultura vertical es un concepto que está en su infancia técnica pero que promete para las ciudades futuras. La investigación adicional puede ayudar a continuar con esta idea. Esto incluye desarrollar diseños multifuncionales con aportes de ingenieros, arquitectos y proveedores de tecnología agrícola vertical simultáneamente para ayudar a diseñar estructuras futuras que puedan adaptarse a las necesidades del siglo XXI, desarrollando programas piloto donde se puedan recopilar y analizar datos en tiempo real para examinar dónde existen oportunidades y barreras en comparación con los productos convencionales, el desarrollo de un modelo de energía más grande que pueda tener más factores en cuenta (ventilación, desperdicio, etc.) y la realización de un estudio tecnoeconómico que incorpora los costos de construcción y mantenimiento. La agricultura vertical tiene potencial en las circunstancias correctas. En esos casos y con un poco más de investigación, el cielo es el límite. (p. 4).

## **Objetivos**

### **General**

Desarrollar un sistema SCADA capaz de optimizar parámetros climáticos para un sistema doméstico de cultivo agrícola de ambiente controlado.

### **Específicos**

* Seleccionar los elementos transductores y actuadores necesarios para diseñar el sistema doméstico que permita medir y manipular parámetros ambientales importantes para la producción de un cultivo agrícola.
* Almacenar y describir los valores actuales de los parámetros ambientales.
* Indicar los valores deseados para los parámetros ambientales.
* Relacionar los valores deseados con los valores actuales.
* Analizar los valores almacenados para hallar valores optimizados para los parámetros ambientales.
* Relacionar los valores optimizados con los valores deseados.

## **Marco teórico**

## **Marco conceptual**

# **CAPÍTULO II - MARCO METODOLÓGICO**

## **Hipótesis**

Mediante la monitorización, control y registro de los parámetros ambientales de un cultivo se podrá utilizar técnicas de algoritmos genéticos y visión artificial para optimizar la producción del mismo en sistemas agrícolas de ambiente controlado.

## **Variables**

### **Variables independientes**

* Tamaño de la planta cultivada
* Color de la planta cultivada
* Temperatura del agua
* Monóxido de oxígeno
* Temperatura de aire
* Volumen de agua
* Intensidad de Luz
* Periodo de inyección de aire fresco
* Periodo de inyección de agua fresca
* Periodo de circulación de aire
* Periodo de circulación de agua

### **Variables dependientes**

* Salida del controlador de temperatura
* Salida del controlador de nivel de agua
* Salida del controlador del periodo de inyección de agua fresca
* Salida del controlador del periodo de inyección de aire fresco
* Salida del controlador del periodo de circulación de aire
* Salida del controlador del periodo de circulación de agua
* Salida del controlador de la intensidad de luz.

Tabla3.1: Variables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Definición conceptual** | **Definición operacional** | **Indicadores** | **Escala de medición** |
| Temperatura de aire | Magnitud física que expresa el grado o nivel de calor del aire, y cuya unidad en el sistema internacional es el kelvin (K). | Temperatura del aire dentro del prototipo de PFC, Medible a través de un sensor y alterable por medio de un sistema de control | °C | 15 - 25 |
| Nivel de agua | Altura de agua | Altura de la solución usada dentro del prototipo de PFC, medible a través de un sensor y alterable por medio de un sistema de control | cm | 0 - 20 |
| Intensidad de luz | Magnitud física que expresa el flujo luminoso emitido por una fuente puntual en una dirección determinada por unidad de ángulo sólido. Su unidad en el Sistema Internacional es la candela | Intensidad de luz dentro del prototipo de PFC y medible a través de un sensor | lx | 0 - 1000 |
| Periodo de injección de aire fresco | Espacio de tiempo hasta la próxima adición de aire | Periodo con la que el prototipo de PFC es recibe aire fresco y alterable por medio de un sistema de control | h | 2 - 4 |
| Periodo de injección de agua fresca | Espacio de tiempo hasta la próxima adición de agua | Periodo con la que el prototipo de PFC es recibe aire fresco y alterable por medio de un sistema de control | h | 3 - 4 |
| Periodo de circulación de aire | Espacio de tiempo hasta la próxima circulación de aire | Periodo con la que el prototipo de PFC es recibe agua fresca y alterable por medio de un sistema de control | h | 4 - 4 |
| Periodo de circulación de agua | Espacio de tiempo hasta la próxima circulación de agua | Periodo con la que circula el agua en el prototipo de PFC y alterable por medio de un sistema de control | h | 5 - 4 |
| Color del cultivo | cualidad de las plantas por la cual impresionan la retina de modo diferente según cómo reflejen los rayos luminosos | Periodo con la que circula el aire en el prototipo de PFC y alterable por medio de un sistema de control | RGB | 0 - 255, 0 - 255, 0 - 255 |
| Área del cultivo | Espacio comprendido por el cultivo | Áreas cultivadas en el prototipo de PFC y medible indirectamente a través de la aplicación de algoritmos de visión artificial sobre una imagen digital | m² | 0 - 0.5 |
| Temperatura del agua | Magnitud física que expresa el grado o nivel de calor del agua, y cuya unidad en el sistema internacional es el kelvin (K). | Temperatura de la solución usada dentro del prototipo de PFC y medible a través de un sensor | °C | 15 - 25 |
| Concentración de monóxido de oxígeno | Magnitud que expresa la cantidad de una sustancia por unidad de volumen | Concentración de monóxido de carbono en el prototipo de PFC y medible a través de un sensor | ppm | 20 - 2000 |
| Salida del controlador de temperatura de aire | Valor resultado del sistema encargado de ejercer el control sobre la temperatura del aire | Valor de salida de un sistema de control en lazo cerrado para variar la temperatura de aire hacia un valor deseado | PWM | 0 - 255 |
| Salida del controlador de nivel de agua | Valor resultado del sistema encargado de ejercer el control sobre el nivel de agua | Valor de salida de un sistema de control en lazo cerrado para variar el nivel de agua hacia un valor deseado | PWM | 0 - 255 |
| Salida del controlador del periodo de inyección de agua fresca | Valor resultado del sistema encargado de ejercer el control sobre el estado de inyección de agua fresca | Valor de salida de un sistema de control en lazo abierto para variar el estado de inyección de agua fresca hacia un valor deseado | V | 0 - 5 |
| Salida del controlador del periodo de inyección de aire fresco | Valor resultado del sistema encargado de ejercer el control sobre el estado de inyección de aire fresco | Valor de salida de un sistema de control en lazo abierto para variar el estado de inyección de aire fresca hacia un valor deseado | V | 0 - 5 |
| Salida del controlador del periodo de circulación de aire | Valor resultado del sistema encargado de ejercer el control sobre el estado de circulación de aire | Valor de salida de un sistema de control en lazo abierto para variar el estado de circulación de aire hacia un valor deseado | V | 0 - 5 |
| Salida del controlador del periodo de circulación de agua | Valor resultado del sistema encargado de ejercer el control sobre el estado de circulación de agua | Valor de salida de un sistema de control en lazo abierto para variar el estado de circulación de agua hacia un valor deseado | V | 0 - 5 |
| Salida del controlador de la intensidad de luz. | Valor resultado del sistema encargado de ejercer el control sobre la intensidad de luz | Valor de salida de un sistema de control en lazo abierto para variar la intensidad de luz hacia un valor deseado | PWM | 0-255 |

## **Metodología**

### **Tipo de estudio**

# Aplicado y exploratorio

### **Diseño**

# Experimental puro.

## **Población y muestra**

### **Población**

# Módulos de software que permitan el registro, supervisión y control de datos.

### **Muestra**

# Módulos de software que permitan registrar, supervisar y controlar parámetros ambientales en sistemas de Agricultura de Ambiente Controlado (CEA).

### **Método de investigación**

# Método heurístico y experimental.

## **Técnicas e instrumentos de recolección de datos**

### **Técnicas**

# Medición de valores proveniente de sensores.

# Revisión de bibliografía especializada para obtención de rangos recomendados para las condiciones climáticas en distintos cultivos agrícolas.

### **Instrumentos**

# Entorno Node.js.

# Software de muestreo y adquisición de datos de elaboración propia.

# Plataforma Arduino.

# Firmware firmata para Arduino.

# Actuadores.

# Sensores.

# Multímetro.

### **Métodos de análisis de datos**

# Aplicación de filtros digitales sobre los datos provenientes de los sensores para la estimación y validación de los modelos de planta respectivos.

# Selección de valores preliminares a partir de valores aleatorios dentro de los rangos de trabajo de los parámetros ambientales de un sistema agrícola de ambiente controlable.

# Utilización de los rangos recomendados en el diseño de un algoritmo genético y visión artificial que permita encontrar valores climáticos más beneficiosos para los cultivos agrícolas.

# **CAPÍTULO III - DESARROLLO Y RESULTADOS**

## **Selección de elementos transductores y actuadores, y diseño de un prototipo de una Personal Food Computer (PFC) como sistema doméstico de cultivo agrícola de ambiente controlado.**

La revisión de bibliografía especializada permitió determinar las siguientes características de trabajo para el sistema:

|  |  |
| --- | --- |
| **CARACTERÍSTICA** | **ESPECIFICACIONES** |
| Humedad relativa | 30 - 90 % |
| Dióxido de carbono | 350 - 1000 ppm |
| Luz | * **Intensidad** 0 - 20 mol/m2-day * **longitudes de onda** 600 a 680 nm (rojo) 380 a 480 nm (azul) |
| Solución nutritiva | * **Mantener** Distribución uniforme Oxigenación * **Conductividad eléctrica** 1.0 - 2.5 dS/m * **Temperatura** 15.5 – 23 °C * **pH** 5.5 - 7.0 |
| Aire | * **Mantener** Distribución uniforme Oxigenación * **Temperatura** 15 – 27 °C |

### **Selección de elementos transductores y actuadores**

El conjunto de elementos listados a continuación, se tomó de un universo de alternativas ofertadas en el mercado peruano, que cumplen con las características requeridas en el cuadro TAL. Dichos elementos serán evaluados por distintos indicadores, cuya finalidad es evaluar la necesidad de su uso en el prototipo, tomando en cuenta los fines de este trabajo. A continuación, se muestran la tabla de indicadores y la tabla de evaluación utilizadas:

|  |  |  |  |
| --- | --- | --- | --- |
| **INDICADOR** | **DESCRIPCIÓN** | **ESCALA** | **PESO** |
| Sustituibilidad | Grado con que el elemento puede ser reemplazado por otro menos complejo con iguales fines demostrativos | 0 a 10 | 3 |
| Precio | Percepción personal cuantitativa del costo  del elemento | 2 |
| Conexiones | Grado de requerimientos necesitados por el elemento para ser añadido al sistema | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ELEMENTO** | **ESPECIFICACIONES TÉCNICAS** | **FACTORES** | | | **RESULTADO** |
| **SUSTITUIBILIDAD** | **CONEXIONES** | **PRECIO** |
| SHT31 | * **Función:** Sensor de humedad y temperatura * **Rangos de trabajo (precisión):** -40° a 125°C (0.2°C) 0 a 100% RH (0.01 %RH) | 7 | 5 | 7 | 6.33 |
| MH-Z19 | * **Función:** Sensor de Dióxido de Carbono * **Rangos de trabajo (precisión):** 0 a 5000ppm (50ppm) * **Condiciones de trabajo:** Precalentamiento de 3min Temperatura de 0 a 50°C Humedad de 0 a 95% HR | 9 | 8 | 10 | 8.83 |
| Cinta LED GROW | * **Función:** Iluminación * **Proporción de colores:** 3 LEDs rojos /c LED azul * **Espectros de luz:**   460nm (azul)  640nm (roja)   * **Potencia**   0.24W/LED | 3 | 1 | 3 | 2.33 |
| Kit pH | * **Función:** Sensor de Ph * **Rangos de trabajo (precisión):** 0 a 14 (0.5) | 5 | 8 | 10 | 6.83 |
| Kit TEC1-12706 | * **Función:** Refrigeración o calefacción * **Corriente de trabajo:**   < 6A   * **Potencia nominal:**   50 a 70W   * **Temperatura de trabajo:**   -30 a 70ºC | 7 | 5 | 10 | 6.83 |
| ME2-O2 | * **Función:** Sensor de Oxígeno * **Rangos de trabajo (precisión):** 0% - 25% * **Condiciones de trabajo:** Concentración medible < 30% Sensibilidad de 0.1 a 0.3 mA Tiempo de respuesta < 15s Temperatura de 20° a 50°C Humedad de 0 a 99% HR | 5 | 8 | 10 | 6.83 |
| MDC-PUMP | * **Función:** Bomba de agua, aceite o gasolina * **Condiciones de trabajo:** Corriente máxima de 350mA Potencia de 4.8W Caudal máximo de 240L/H Columna de agua < 3m Ruido < 40dB Diseño sumergible Dimensiones de 38\*36\*25mm Temperatura de 0 a 60°C | 1 | 1 | 4 | 1.5 |
| SEN-DS18B20 | * **Función:** Sensor acuático de temperatura * **Rango de Trabajo (Precisión):** -55 a 125℃ (0.5°C) | 8 | 2 | 5 | 5.5 |
| foxconn DC Brushless Fan | * **Función:** Ventilador * **Condiciones de trabajo:** -40 a 70°C * **Velocidad nominal:**   19000 RPM   * **Flujo de aire:**   25.6CFM | 1 | 1 | 1 | 1 |
| Nutrientes | Soluciones nutritivas A y B | 10 | 0 | 7 | 6.17 |
| MQ7 | * **Función:** Sensor de Monóxido de Carbono * **Rangos de trabajo (precisión):** 0 a 2000ppm (20ppm) * **Condiciones de trabajo:** Precalentamiento de 3min Temperatura de -20 a 50°C Humedad de 0 a 95% HR | 1 | 2 | 1 | 1. 33 |
| Windshield Washer Pump | Bomba limpiaparabrisas 12VDC | 1 | 1 | 2 | 1.17 |
| LM35 | * **Función:** Sensor analógico de temperatura * **Rango de trabajo (Precisión):** -55 a 150℃ (0.5°C) * **Pendiente**   10mV / ºC | 1 | 2 | 7 | 2.33 |
| ACT-HUM | * **Función:** Humidificador ultrasónico * **Condiciones de operación:** Voltaje de Operación: 24V DC Consumo de corriente: 800mA Diseño sumergible | 2 | 8 | 7 | 4.83 |
| HC-SR04 | * **Función:** Sensor ultrasónico * **Rango de trabajo (Precisión):** 2 a 450cm (3mm) Ángulo de apertura: 15° Corriente de trabajo de 15mA | 1 | 3 | 1 | 1.67 |
| SB-348A  Air Pump | Bomba de aire 220VAC | 1 | 1 | 5 | 1.67 |

La presencia de los elementos con una puntuación menor a cinco (5) es factible y altamente necesarios, por lo tanto, serán considerados en el diseño del prototipo. Los demás serán reemplazados por elementos o sistemas sencillos que ofrezcan iguales características demostrativas para este trabajo.

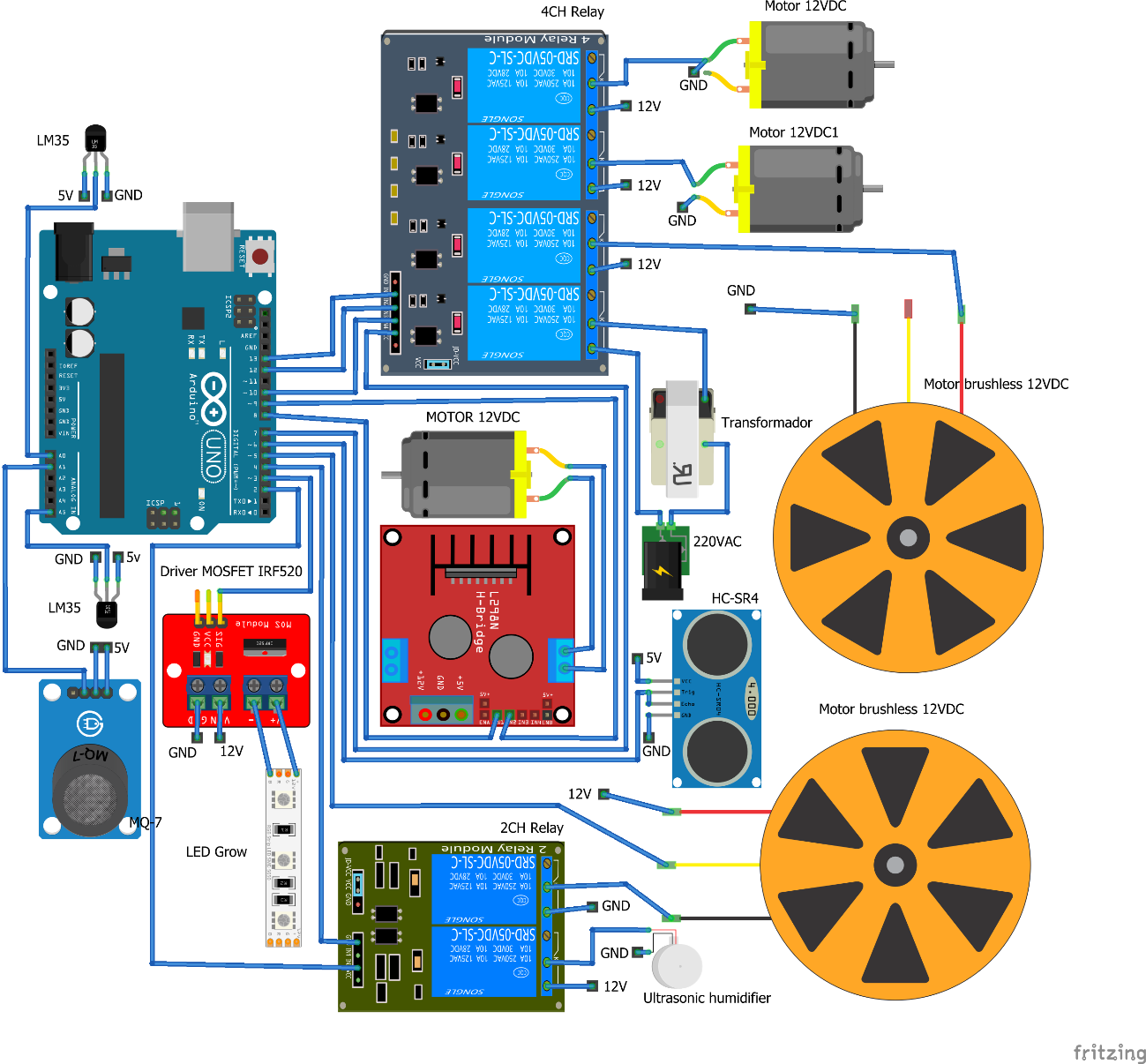
### **Diseño del prototipo**

Con los transductores y actuadores seleccionados, es necesario incorporar los elementos que permitan su manipulación o automatización. Así también, como los elementos que permitan la conformación completa de la planta.

A continuación, se muestra la lista completa de los elementos del sistema electrónico del prototipo:

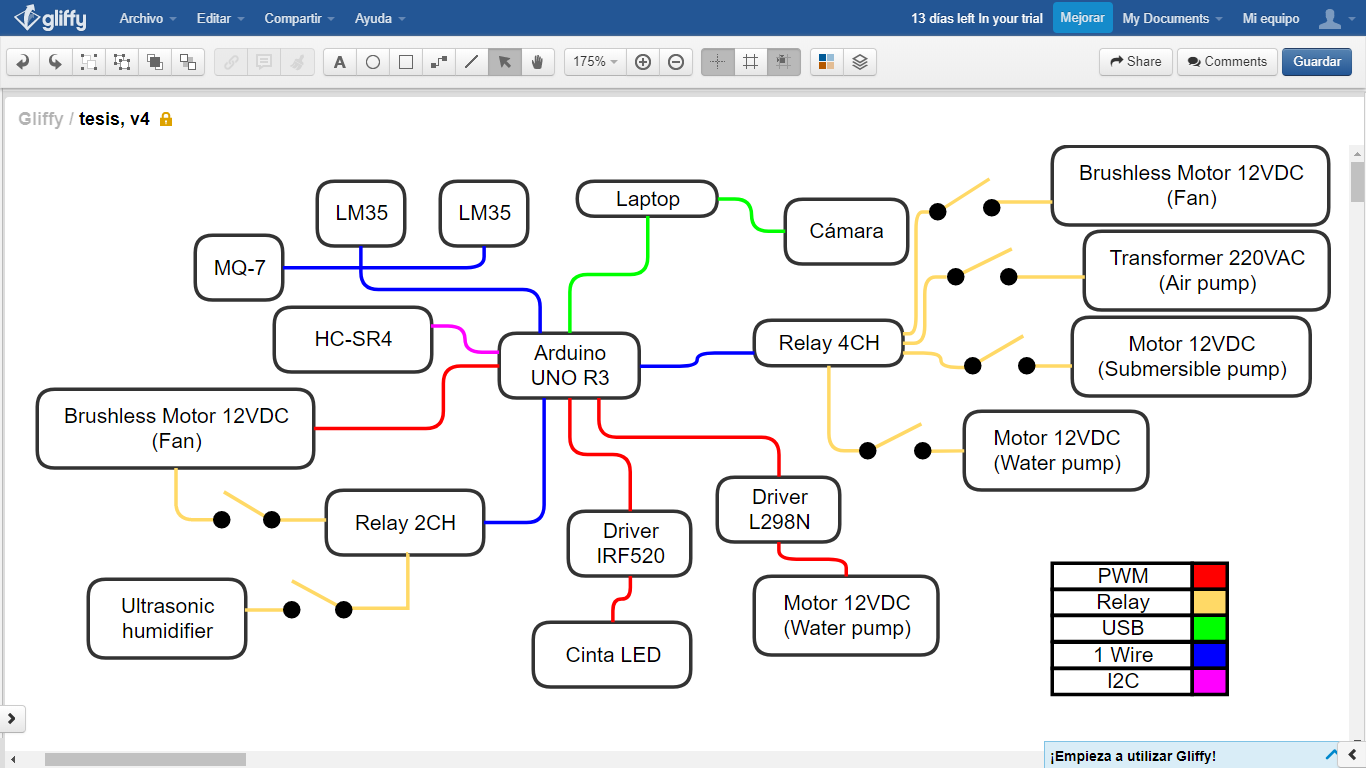
|  |  |  |  |
| --- | --- | --- | --- |
| **HARDWARE** | | | |
| **SISTEMA ELECTRÓNICO** | | | |
| **N°** | **ELEMENTO** | **REQUERIMIENTO** | **DESCRIPCIÓN** |
| 1 | Cinta LED GROW | 2 metros | Emisión de luz apropiada para fotosíntesis. |
| 2 | Bomba de Agua MDC-PUMP | 1 unidad | Recirculación de la solución nutritiva. |
| 3 | foxconn DC Brushless Fan | 2 unidades | Agregación y recirculación de aire frío. |
| 4 | Sensor MQ7 | 1 unidad | Detección de monóxido de carbono. |
| 5 | Windshield Washer Pumps | 2 unidades | Provisión de agua fresca y nutrientes. |
| 6 | LM35 | 2 unidades | Detección de temperatura. |
| 7 | HC-SR04 | 1 unidad | Detección de nivel de líquido. |
| 8 | AIR PUMP SB-348A | 1 unidad | Provisión de aire fresco. |
| 9 | ACT-HUM | 1 unidad | Aumento de humedad relativa del aire. |
| 10 | Driver Mosfet IRF520 | 1 unidad | Control de la cinta LED GROW por PWM. |
| 11 | Módulo Relay 4CH 5VDC | 1 unidad | Control ON / OFF de 4 elementos. |
| 12 | Módulo Relay 2CH 5VDC | 1 unidad | Control ON / OFF de 2 elementos. |
| 13 | Driver Puente H DRV-L298N | 1 unidad | Manipulación de bomba de agua por PWM. |
| 14 | Arduino Uno R3 | 1 unidad | Recepción, entrega y transmisión de señales de control y de sensores. |
| 15 | Laptop | 1 unidad | Servidor del software. |
| 16 | Webcam | 1 unidad | Toma de fotos diarias. |
| 17 | Fuente de poder 240W | 1 unidad | Provisión de la energía para el prototipo. |

Con los componentes mencionados anteriormente, el sistema electrónico de control que se implementó es el siguiente.

****

**FIGURA 1: Esquema del sistema electrónico de control**

Todos los tipos de conexiones utilizadas en el sistema electrónico es el siguiente:



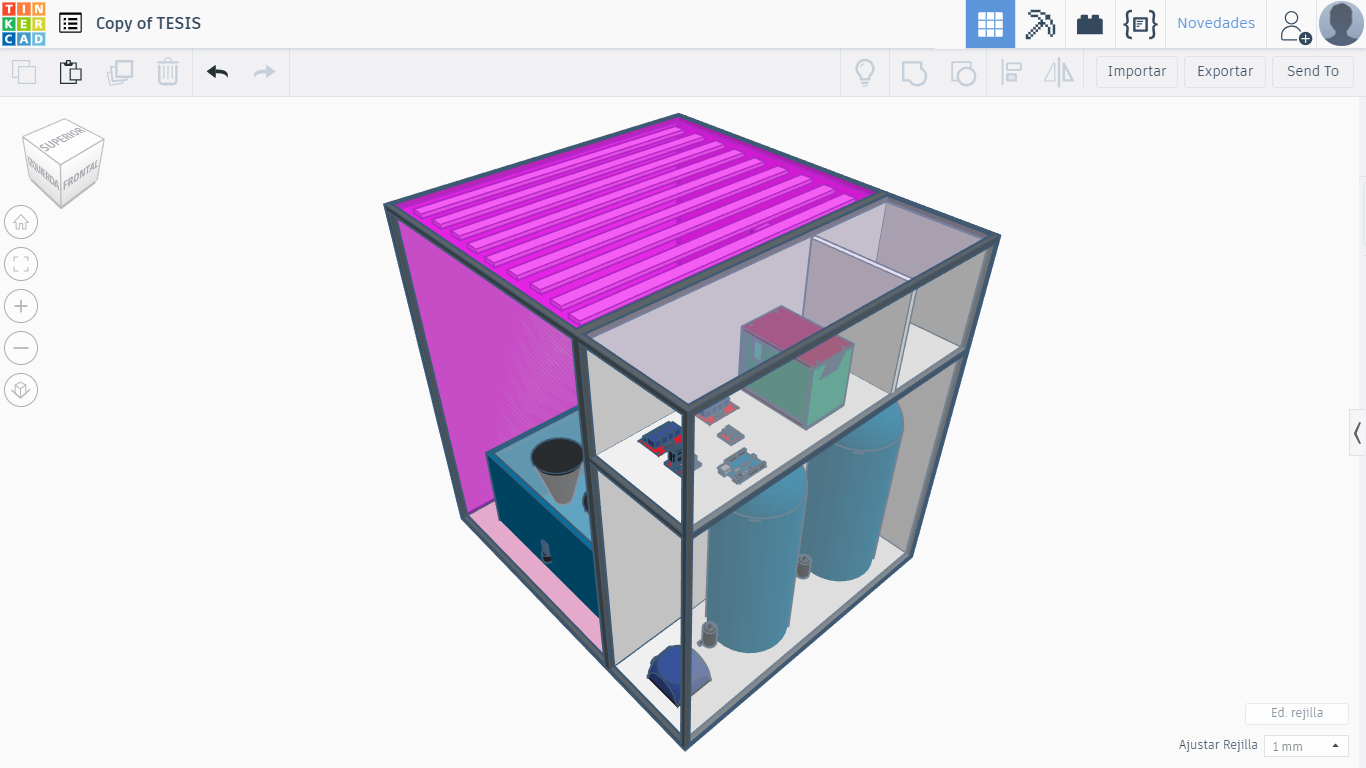
**FIGURA: Esquema general de conexiones del sistema electrónico**

En el caso de la estructura y los elementos auxiliares del prototipo, las dimensiones dependerán del volumen de aire impulsado por los ventiladores en un determinado tiempo. Puesto que este valor es de 25.6 pies3/minuto que equivalen a 0.725 m3/minuto; y tomando en cuenta que alrededor de un minuto es un tiempo adecuado para llenar el volumen del prototipo, este no debe exceder de 0.725 m3.

Las dimensiones de 0.8 m de ancho, 0.65 m de largo y 0.9 m de altura permitirán cultivar una gran cantidad de tipos de cultivos de distintas alturas en un área prudente, sin comprometer las capacidades de los ventiladores. Entonces, puesto que el espacio destinado para el cultivo, según las dimesiones mencionadas anteriormente, es un volumen de 0.468 m3, los ventiladores serán capaces de impulsar ese volumen de aire en aproximadamente:

A continuación, se muestra los elementos necesarios para el diseño de la estructura física del prototipo:

|  |  |  |  |
| --- | --- | --- | --- |
| **HARDWARE** | | | |
| **ESTRUCTURA FÍSICA** | | | |
| 18 | Armadura de perfiles de acero | Ángulos laminados: 25 cm de largo x 2 90 cm de largo x 6 80 cm de largo x 12 | Medidas que cubren el espacio requerido para el cultivo como el espacio para la electrónica y demás elementos. |
| 19 | Teknopor | 6 planchas (120 x 80 cm2) | Material seleccionado por sus propiedades adiabáticas para recubrir la estructura. |
| 20 | Cartulina | 6 planchas (120 x 80 cm2) | Se usará para recubrir el teknopor por su parte exterior. |
| 21 | Papel aluminio | 15 metros | Se usará para recubrir el teknopor por su parte interior para reflectar la luz. |
| 22 | Hielo | 3 kilogramos | Se usará para proporcionar al prototipo de una fuente de aire frío. |
| 23 | Agua | 10 litros | Se usará para simular la fuentes de nutriente y de agua. |
| 24 | Mangueras | 3 metros | Necesarias para las salidas de todas las bombas. |
| 25 | Vasos negros de plástico | 6 unidades | Recipientes para las semillas. |

90 cm

25 cm

65 cm

25 cm

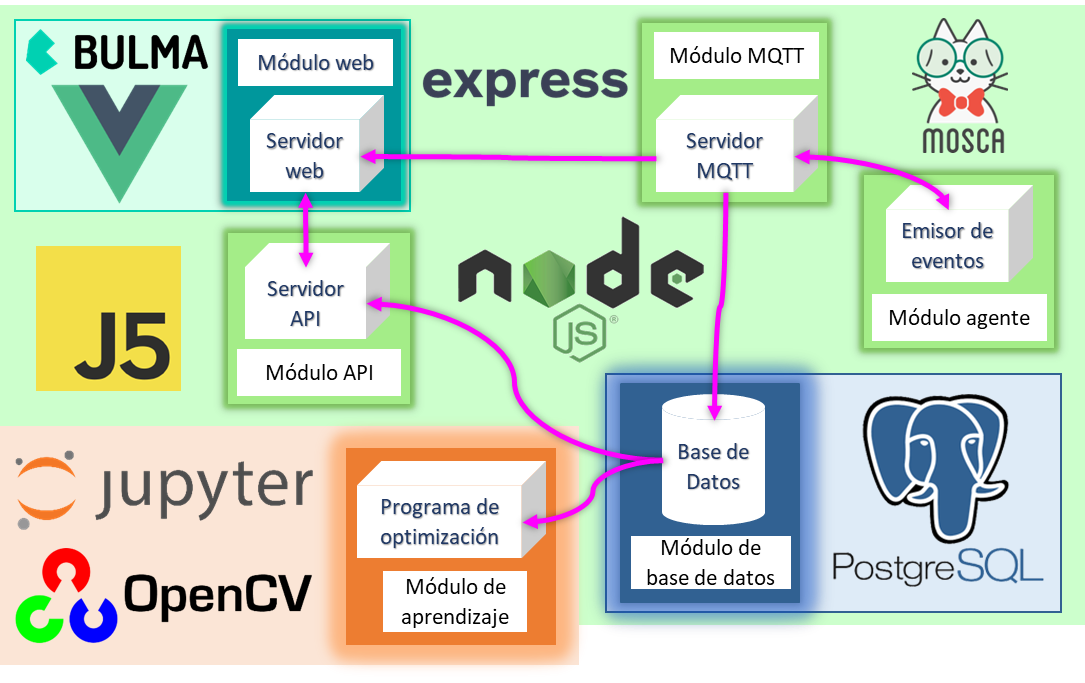
80 cm

25 cm

**Figura: Vistas auxiliares del diseño 3D del hardware del sistema**

En el caso del SCADA inteligente, el diseño estará basado en las siguientes tecnologías:

|  |  |  |
| --- | --- | --- |
| **SOFTWARE** | | |
| **N°** | **ELEMENTO** | **DESCRIPCIÓN** |
| 26 | Node.js | Entorno de ejecución para JavaScript que ofrece una gran cantidad de opciones para desarrollo de software. |
| 27 | Bulma | Framework CSS open source usado para proporcionar los estilos CSS a la web del proyecto. |
| 28 | Arduino IDE | Entorno de desarrollo integrado para placas Arduino. Se usará la implementación del protocolo firmata. |
| 29 | Linux | Sistema operativo open source basado en UNIX. |
| 30 | Jupyter  Notebook | Aplicación web que permite crear y compartir documentos que contienen código fuente, ecuaciones, visualizaciones y texto explicativo. Permitirá desarrollar el programa de visón artifical y algoritmos genéticos. |
| 31 | OpenCV | Librería Open Source de visión artificial. |
| 32 | MatLab | Entorno para análisis iterativo y procesos de diseño. Se usará para hallar parámetros de control. |

Se presenta la arquitectura del SCADA:

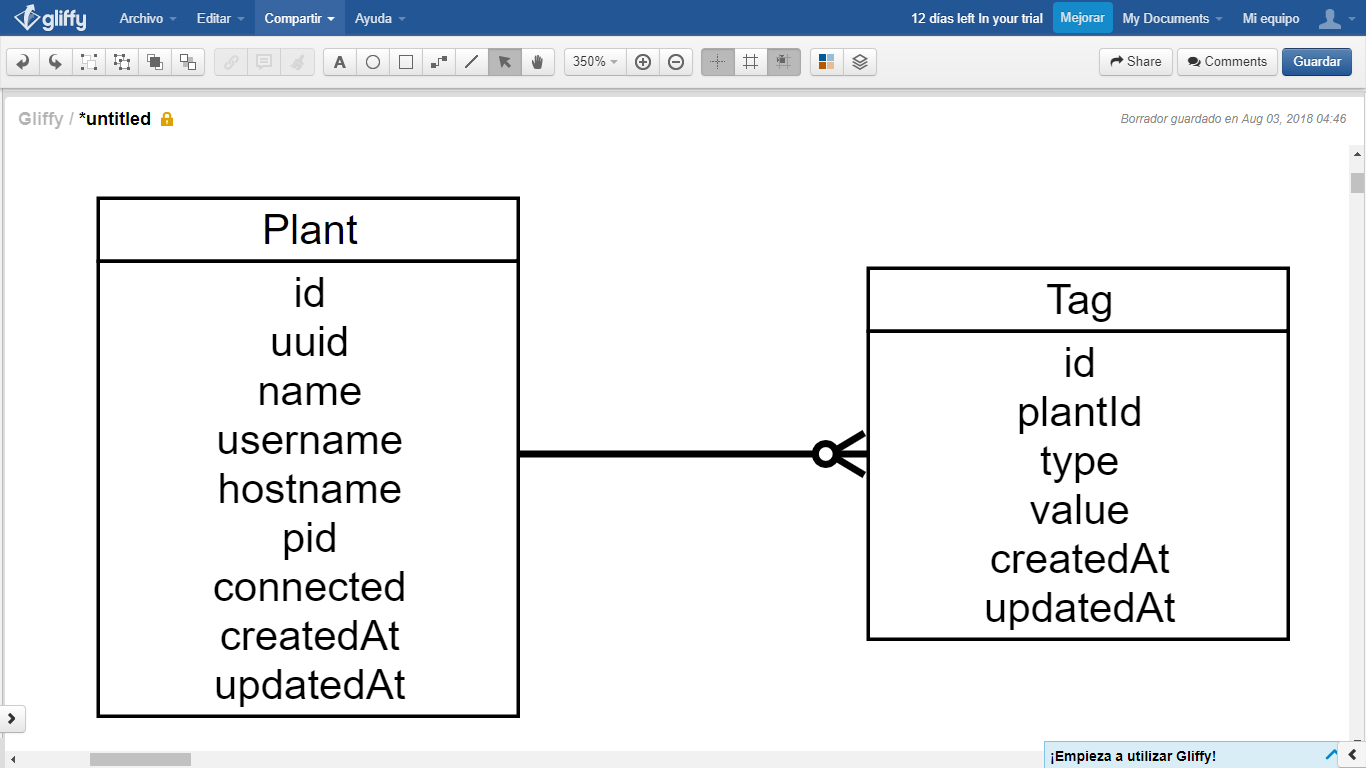
**FIGURA: Arquitectura del SCADA inteligente y disposición de las tecnologías seleccionadas.**

## **Almacenamiento y descripción de los valores actuales de los parámetros ambientales.**

Antes de almacenar los valores filtrados y los estados de algunos de los actuadores que, en conjunto, conforman las condiciones ambientales del sistema, se creó una base de datos relacional SQL usando PostgreSQL. Para el modelamiento de la base de datos se tuvo en cuenta dos entidades y los siguientes atributos:

|  |  |
| --- | --- |
| **ENTIDAD** | **DESCRIPCIÓN** |
| Plant | * Conceptualiza a cada sistema doméstico de cultivo agrícola. * **Atributos:** *id*: identificador incremental por defecto *uuid*: identificador aleatorio legible  *name*: nombre del tipo de cultivo *username*: nombre del administrador *hostname*: nombre del servidor *pid*: identificador de proceso *connected*: estado de conexión *createdAt*: fecha de creación *updateAt*: fecha actual |
| Tag | * Conceptualiza a cada condición ambiental que puede transmitir el sistema. * **Atributos:** *id*: identificador incremental por defecto *plantId*: identificador de la planta *type*: nombre de la condición ambiental *value*: valor de la condición ambiental *createdAt*: fecha de creación *updateAt*: fecha de actualización |

Una entidad Plant puede tener *cero o muchas* condiciones ambientales y una entidad Tag debe pertenecer *obligatoriamente a una* entidad Plant. El modelo Entidad – Relación de la base de datos queda entonces:



Para interactuar con la base de datos, se utilizó el ORM para Node.js: Sequelize. Con esta librería se creó un módulo de base de datos para el SCADA inteligente. La estructura de archivos del módulo de base de datos es la siguiente:

La función de cada archivo se detalla a continuación:

|  |  |
| --- | --- |
| **ARCHIVO** | **DESCRIPCIÓN** |
| lib/plant.js | Entrega un objeto con varios métodos para la entidad Plant. |
| lib/tag.js | Entrega un objeto con varios métodos para la entidad Tag. |
| db.js | Entrega a Sequelize una configuración predeterminada. |
| models/plant.js | Define la entidad Plant en la base de datos.  **Métodos:**   * **createOrUpdate**: crea o actualiza a la entidad Plant * **findById**: muestra la entidad Plant con un Id específico * **findByUuid**: muestra la entidad Plant con un Uuid específico * **findAll**: muestra todas las entidades Plant * **findConnected**: muestra las entidades Plant con *connected: true* * **findByUsername**: muestra las entidades Plant con un Username específico |
| models/tag.js | Define la entidad Tag en la base de datos.  **Métodos:**   * **create**: crea la entidad Tag para una entidad Tag * **findByAgentUuid**: muestra la entidad Plant a la que pertenece la entidad Tag. * **findByTypeAgentUuid**: muestra la entidad Tag por tipo y el Uuid de Plant |
| index.js | Configura la base de datos y devuelve los objetos de los entidades. |
| setup.js | Limpia la base de datos sin perder la configuración y ni el modelamiento. |

El sistema SCADA debe ser escalable y tener la capacidad de interactuar con uno o varios clientes (Entidades Plant) en una o varias estaciones de cultivo; es decir, debe funcionar como el servidor central para manejar todos los eventos. Para lograr esto, es conveniente el uso de un protocolo conveniente para Internet de las Cosas como MQTT, que permite utilizar el modelo Pub/Sub y está optimizada para cuando la calidad de la señal de internet es baja.

En este trabajo, se utilizó el bróker MQTT de Node.js: **Mosca.** Mosca permite implementar el modelo Pub/Sub en MQTT. La estructura de archivos del módulo MQTT es la siguiente:

La función de cada archivo se detalla a continuación:

|  |  |
| --- | --- |
| **ARCHIVO** | **DESCRIPCIÓN** |
| server.js | Gestiona distintos eventos (mensajes) entre el servidor MQTT y los clientes (Entidades Plant) **Eventos:**   * **'ready':** Conexión a la base de datos cuando el servidor está listo, obteniendo las entidades Plant y Tag * **'clientConnected'**: Guarda el Id de la entidad Plant. * **'clientDisconnected'**: Excluye de la lista a la entidad Plant. * **'agent/disconnected'**: Permite hacer debug * **'published'**: Gestiona los eventos agent/disconnected y agent/message * **'agent/message'**: Crea o actualiza el estado de las entidades Plant y Tag |
| utils.js | Una función que parsea los mensajes para darles formato JSON si lo necesitan. |

El servidor MQTT permite la interacción con la base de datos a través respuestas a eventos, por lo tanto, fue necesario implementar un módulo cliente para el servidor MQTT. Para esto se usó el cliente MQTT para Node.js: **MQTT.js**. Este módulo cliente se denominó como el módulo Plant, que conceptualiza a un cliente en una estación y permite emitir las condiciones ambientales hacia el servidor MQTT. La estructura de archivos del módulo Plant es la siguiente:

La función de cada archivo se detalla a continuación:

|  |  |
| --- | --- |
| **ARCHIVO** | **DESCRIPCIÓN** |
| index.js | Permite crear un objeto que envíe eventos hacia el servidor MQTT. **Métodos:**   * ***addMetric***: agrega una entidad Tag a su entidad Plant. * ***removeMetric***: borra una entidad Tag de su entidad Plant. * ***connect***: se suscribe a los temas de los eventos del servidor MQTT para recibir mensajes de otros clientes, y envía un mensaje cada cierto tiempo con información completa de las entidades Plant y Tag. * ***disconnect***: desconecta el cliente, termina el envió de mensajes y las suscripciones. |
| utils.js | Una función que parsea los mensajes para darles formato JSON si lo necesitan. |

Para que el sistema SCADA sea capaz de responder a peticiones HTTP de un cliente, fue necesario implementar una API que reciba dichas peticiones, se comunique con la base de datos y responda adecuadamente, además de proporcionar una capa de seguridad para evitar accesos no deseados. La estructura de archivos del módulo api es la siguiente:

La función de cada archivo se detalla a continuación:

|  |  |
| --- | --- |
| **ARCHIVO** | **DESCRIPCIÓN** |
| api.js | Define las rutas (Endpoints) de las peticiones a través de una capa de seguridad.  **Endpoints:**   * **'\*':** Conexión a la base de datos para recibir las entidades Plant y Tag. * **'/plants':** Entrega de las todas o una de las entidades Plant, dependiendo de los permisos del usuario. * **'/plant/:uuid'**: Entrega de la entidad Plant con un Uuid específico. * **'/tags/:uuid':** Entrega de las entidades Tag de la entidad Plant con un Uuid específico. * **'/tags/:uuid/:type':** Entrega de las entidades Tag con un Type específico, de la entidad Plant con un Uuid específico. |
| auth.js | Función que permite implementar una capa de seguridad bajo el estándar JSON Web Token por medio de la librería para Node.js: jsonwebtoken**.** |
| server.js | Implementación del servidor HTTP de la api. |

Para hacer peticiones desde un cliente (Un navegador web, por ejemplo) hacia la API, es necesario implementar un servidor HTTP intermediario entre el cliente y la API. Para habilitar una comunicación full dúplex en tiempo real entre cliente y servidor, se usó la implementación de websockets con la librería **Socket.IO** de JavaScript y para mostrar los valores cambiantes de las condiciones ambientales, se usó una interfaz de usuario con un framework reactivo que implemente un *virtual document* que se actualice automáticamente con los datos actuales de las condiciones ambientales.

Con los siguientes fragmentos de código en Vue.js se pudo mostrar los valores de las condiciones ambientales del sistema en la interfaz de usuario (Con dos valores decimales):

**Descripción de la temperatura del aire**

<p class="title is-size-5">Air Temperature</p>

<p class="subtitle tag is-warning">{{ Math.round(rightNowElement \* 10) / 10 }} C°</p>

**Descripción del nivel de solución nutritiva**

<p class="title is-size-5">Nutritive Solution Level</p>

<p class="subtitle tag is-warning">{{ Math.round(rightNowElement \* 10) / 10 }} cm</p>

**Descripción de la intensidad de la luz**

<p class="title is-size-5">Light Intensity</p>

<p class="subtitle tag is-warning">{{ Math.round(rightNowElement\*65535/255) }} lx</p>

**Descripción del estado de la inyección de aire fresco**

<p class="title is-size-5">Fresh Air</p>

<p class="subtitle tag is-warning">{{ rightNowElement }}</p>

**Descripción del estado de la inyección de solución nutritiva**

<p class="title is-size-5">Fresh Nutritive Solution</p>

<p class="subtitle tag is-warning">{{ rightNowElement }}</p>

**Descripción del estado de circulación del aire**

<p class="title is-size-5">Air Circulation</p>

<p class="subtitle tag is-warning">{{ rightNowElement }}</p>

**Descripción del estado de circulación de la solución nutritiva**

<p class="title is-size-5">Nutritive Solution Circulation</p>

<p class="subtitle tag is-warning">{{ rightNowElement }}</p>

**Descripción de la temperatura de la solución nutritiva**

<p class="title is-size-5">Nutritive Solution Temperature</p>

<p class="subtitle tag is-warning">{{ Math.round(rightNowElement \* 10) / 10 }} C°</p>

**Descripción de la cantidad de monóxido de oxígeno**

<p class="title is-size-5">Oxygen Monoxide</p>

<p class="subtitle tag is-warning">{{ Math.round(rightNowElement\*10) / 10 }} ppm</p>

A continuación, se muestra la estructura de archivos del módulo web:

La función de cada archivo se detalla a continuación:

|  |  |
| --- | --- |
| **ARCHIVO** | **DESCRIPCIÓN** |
| config.js | Contenedor del identificador URL del servidor de la API y el token de autenticación. |
| proxy.js | Define rutas (Las mismas definidas en el módulo api) y el cuerpo de la solicitud para comunicarse con la API de forma indirecta. |
| server.js | Implementa el servidor web, la comunicación en tiempo real con websockets, el uso de una entidad Plant con Jhonny-Five para comunicarse con el Arduino y el envío de los valores de las condiciones ambientales para ser vistos en la interfaz gráfica. |
| utils.js | Función que toma un mensaje de un EventEmitter de una entidad Plant y lo transforma a un mensaje más limpio. Toma los eventos de la entidad Plant y los envía al objeto. Socket. |
| /client | **Archivos:**   * **'\*':** Conexión a la base de datos para recibir las entidades Plant y Tag. * **'/plants':** Entrega de las todas o una de las entidades Plant, dependiendo de los permisos del usuario. * **'/plant/:uuid'**: Entrega de la entidad Plant con un Uuid específico. * **'/tags/:uuid':** Entrega de las entidades Tag de la entidad Plant con un Uuid específico. * **'/tags/:uuid/:type':** Entrega de las entidades Tag con un Type específico, de la entidad Plant con un Uuid específico. |
| /public | Carpeta contenedora de las imágenes, estilos y los archivos generados por el framework Vue.js. |

Hasta este punto, el SCADA ya está habilitado para almacenar y describir las condiciones ambientales. Estas condiciones ambientales, captadas por la placa Arduino a través de los sensores y del estado de los actuadores, deben ser enviados a laptop para ser usados por el SCADA del sistema. Esta transmisión será posible con cargar a la placa Arduino una versión apropiada del protocolo Firmata. Puesto que se usará un sensor HC-SR04, será necesario cargar la versión denominada **PingFirmata**, que es una modificación de la versión standard del protocolo con soporte para la extensión PING\_READ. Con el protocolo cargado, ya es posible establecer la comunicación con la placa Arduino desde un programa en la laptop. Para la construcción del programa se usó la plataforma de Robótica & IoT de JavaScript para Node.js: **Jhonny-Five**. De aquí en adelante, se mencionará a la placa Arduino, simplemente como Arduino.

Para poder usar los datos de las condiciones ambientales, con Jhonny-Five se necesitó usar las siguientes clases:

|  |  |
| --- | --- |
| **CLASE** | **DESCRIPCIÓN** |
| Board | Conexión y control del Arduino por el puerto serie. |
| Led | Manipulación de un led conectado al Arduino. |
| Motor | Manipulación de un driver de motor conectado al Arduino. |
| Pin | Otorgamiento a cualquier pin del Arduino de una específicación. |
| Proximity | Lectura de los datos de un sensor de proximidad conectado al Arduino. |
| Relay | Manipulación de un relay conectado al Arduino. |
| Sensor | Lectura de los datos de un sensor analógico o digital conectado al Arduino. |
| Thermometer | Lectura de los datos de un sensor de temperatura conectado al Arduino. |

Por el posible ruido del sistema o la baja precisión de cada sensor, fue necesario usar filtros digitales sobre los valores captados en el programa. En todos los casos se implementó un filtro digital paso bajo de primer orden:

La forma general en el dominio de Laplace de un filtro paso bajo de primer orden es:

Para implementar el filtro en un microcontrolador, se debe trabajar con la convolución del bloque del filtro, con un bloque de muestreo y retención de orden cero para evitar señales atrasadas:

Para poder utilizar el filtro en el programa, es necesario cambiar del dominio de Laplace al dominio Zeta:

Escribiendo la ecuación de la señal en tiempo discreto queda:

Y en específico con la frecuencia de corte y un tiempo de muestreo queda:

Para este trabajo, se incluyó un emisor de eventos (Entidad Plant) del módulo plant en el servidor web para que, al manipular ciertas variables desde el navegador, a su vez se afecten los mensajes del emisor de eventos. Por lo tanto, en el servidor web se realizó toda la implementación de las lecturas de los sensores y de los comandos de los actuadores con Jhonny-five.js.

Se implementó un objeto Plant para conceptualizar a un cultivo y nueve eventos relacionados con las condiciones ambientales captadas por el sistema. El fragmento de código para lo anterior se muestra a continuación:

const plant = new RaphiAgent({

name: 'Lettuce',

username: 'Irvin',

interval: 100

})

plant.addMetric('AirTemperature', () => {

return airTempOut

})

plant.addMetric('TankLevel', () => {

return tnkLevelOut

})

plant.addMetric('LightIntensity', () => {

return usrValLux0

})

plant.addMetric('NutriSolTemperature', () => {

return nutriSolTempOut

})

plant.addMetric('OxygenMonoxide', () => {

return coOut

})

plant.addMetric('FreshAir', () => {

return usrFshAir0

})

plant.addMetric('FreshNutriSol', () => {

return usrFshNutriSol0

})

plant.addMetric('AirCirculation', () => {

return usrRndAir0

})

plant.addMetric('NutriSolCirculation', () => {

return usrRndNutriSol0

})

plant.connect()

Se necesitó implementar filtros digitales para tres valores de las condiciones ambientales en el servidor web con jhonny-five.js. Lo anterior se muestra en los siguientes fragmentos de código:

**Valores de la señal de la temperatura de la solución nutritiva**

const nutriSolTemperature = new five.Thermometer({

controller: "LM35",

pin: "A0",

freq: 25

})

let nutriSolTemp1 = 0

nutriSolTemperature.on("data", function () {

let y0 = this.celsius \* 0.0609 + y1 \* 0.9391

nutriSolTempOut = nutriSolTemp0

nutriSolTemp1 = nutriSolTemp0

})

**Valores de la señal de la temperatura del aire**

const airTemperature = new five.Thermometer({

controller: "LM35",

pin: "A5",

freq: 25

})

let airTemp1 = 0

airTemperature.on("data", function () {

let airTemp0 = this.celsius \* 0.0609 + airTemp1 \* 0.9391

airTempOut = airTemp0

airTemp1 = airTemp0

})

**Valores de la señal del nivel del agua**

new five.Pin({

pin: 6,

type: "digital"

})

const proximity = new five.Proximity({

controller: "HCSR04",

pin: 6

})

let tnkLevel1 = 0

proximity.on("data", async function () {

let tnkLevel0 = this.cm \* 0.0609 + tnkLevel1 \* 0.9391

tnkLevelOut = 12.1 - tnkLevel0

tnkLevel1 = tnkLevel0

})

El código completo de los módulos implementados se puede encontrar en los anexos.

## **Indicación de los valores deseados para los parámetros ambientales.**

Para poder indicar los Setpoints para las condiciones ambientales controlables del sistema, se implementaron etiquetas *inputs* y *form* en la interfaz de usuario que desencadenan eventos que ejecutan funciones en el servidor con los valores indicados por el usuario. Se muestran los fragmentos de código que permiten lograr esto.

En el archivo *server.js*:

**Eventos de socket.io que recibe el servidor**

socket.on('stateSubmit', (data) => {

let { state, option } = data

switch (option) {

case 'fa':

usrFshAir0 = state

break

case 'fw':

usrFshNutriSol0 = state

break

case 'ra':

usrRndAir0 = state

break

case 'rw':

usrRndNutriSol0 = state

break

default:

break

}

})

socket.on('valueSubmit', (data) => {

let { value, option } = data

switch (option) {

case 'temp':

usrAirTemp0 = value

airTempSp = usrAirTemp0

break

case 'level':

usrTnkLevel0 = value

tnkLevelSp = usrTnkLevel0

break

case 'lux':

usrValLux0 = value

luxSp = usrValLux0

break

default:

break

}

})

En el archivo *metric.vue*:

**Eventos de socket.io que genera el usuario en el cliente**

setState(option) {

const { socket } = this

this.state = this.state ? false : true

socket.emit('stateSubmit', { state: this.state, option: option })

},

setValue(option) {

const { socket } = this

switch (option) {

case 'temp':

this.value = this.$temp.value

break

case 'level':

this.value = this.$level.value

break

case 'lux':

this.value = this.$lux.value

break

default:

break

}

socket.emit('valueSubmit', { value: this.value, option: option })

}

**Indicación de la temperatura del aire:**

<div class="control">

<input id="temp" class="input is-success" type="number" value="0">

</div>

<div class="control">

<button class="button is-success" v-on:click="setValue('temp')"

type="submit" name="action">Set</button>

</div>

**Indicación del nivel de agua en el tanque**

<div class="control">

<input id="level" class="input is-success" type="number" value="0">

</div>

<div class="control">

<button class="button is-success" v-on:click="setValue('level')"

type="submit" name="action">Set</button>

</div>

**Indicación de la intensidad de luz**

<div class="control">

<input id="lux" class="input is-success" type="number" value="0">

</div>

<div class="control">

<button class="button is-success" v-on:click="setValue('lux')"

type="submit" name="action">Set</button>

</div>

**Indicación del estado de inyección de aire fresco**

<div class="control" v-show="!automatic">

<button v-show="!automatic" v-on:click="setState('fa')"

class="button is-success" type="submit"

name="action">Toggle</button>

</div>

**Indicación del estado de inyección agua fresca**

<div class="control" v-show="!automatic">

<button v-show="!automatic" v-on:click="setState('fw')"

class="button is-success" type="submit"

name="action">Toggle</button>

</div>

**Indicación del estado de circulación del aire**

<div class="control" v-show="!automatic">

<button v-show="!automatic" v-on:click="setState('ra')"

class="button is-success" type="submit"

name="action">Toggle</button>

</div>

**Indicación del estado de circulación del agua**

<div class="control" v-show="!automatic">

<button v-show="!automatic" v-on:click="setState('rw')"

class="button is-success" type="submit"

name="action">Toggle</button>

</div>

## **Relación de los valores deseados con los valores actuales.**

La indicación de los valores deseados permite que se desencadenen eventos en el servidor que ejecutan funciones. Estas funciones son las implementaciones de controladores PI, On – Off y por PWM que reciben los valores indicados por el usuario, y en el caso de los controladores PI, las funciones de control se ejecutan cada 100ms.

**Controlador On – Off para la inyección de aire fresco**

const fshAir\_relay = new five.Relay({

pin: 7,

type: "NC"

})

fshAir\_relay.off()

setInterval(() => {

if (usrFshAir0 != usrFshAir1) {

if (usrFshAir0) {

fshAir\_relay.on()

usrFshAir1 = usrFshAir0

} else {

fshAir\_relay.off()

usrFshAir1= usrFshAir0

}

}

}, 300)

**Controlador On – Off para la inyección fresca de solución de nutriente**

new five.Pin({

pin: 10,

type: "digital"

})

const fshNutriSol\_relay = new five.Relay({

pin: 10,

type: "NC"

})

fshNutriSol\_relay.off()

setInterval(() => {

if (usrFshNutriSol0 != usrFshNutriSol1) {

if (usrFshNutriSol0) {

fshNutriSol\_relay.on()

usrFshNutriSol1 = usrFshNutriSol0

} else {

fshNutriSol\_relay.off()

usrFshNutriSol1 = usrFshNutriSol0

}

}

}, 300)

**Controlador On – Off para la circulación de aire**

const rndAir\_relay = new five.Relay({

pin: 12,

type: "NC"

})

rndAir\_relay.off()

setInterval(() => {

if (usrRndAir0 != usrRndAir1) {

if (usrRndAir0) {

rndAir\_relay.on()

usrRndAir1 = usrRndAir0

} else {

rndAir\_relay.off()

usrRndAir1 = usrRndAir0

}

}

}, 300)

**Controlador On – Off para la inyección de aire fresco**

const rndNutriSol\_relay = new five.Relay({

pin: 13,

type: "NC"

})

rndNutriSol\_relay.off()

setInterval(() => {

if (usrRndNutriSol0 != usrRndNutriSol1) {

if (usrRndNutriSol0) {

rndNutriSol\_relay.on()

usrRndNutriSol1 = usrRndNutriSol0

} else {

rndNutriSol\_relay.off()

usrRndNutriSol1 = usrRndNutriSol0

}

}

}, 300)

**Controlador por PWM de la intensidad de luz**

var led = new five.Led(3)

async function luxController(Sp) {

led.brightness( Math.round(Sp) )

}

setInterval(async () => {

if (usrValLux0 != usrValLux1) {

await luxController(luxSp)

}

}, 300)

**Control PI de la temperatura del aire**

board.pinMode(5, five.Pin.PWM)

async function pwmFan (x) {

if (x < 90) {

airTemp\_relay.close()

} else if (x > 255) {

airTemp\_relay.open()

board.analogWrite(5, 255)

} else {

airTemp\_relay.open()

board.analogWrite(5, x)

}

}

let airTemp\_pi0 = 0

let airTemp\_pi1 = 0

let airTemp\_err0 = 0

let airTemp\_err1 = 0

async function airTempPidController (sp) {

airTemp\_err1 = airTemp\_err0

airTemp\_err0 = airTempOut - sp

let airTemp\_pi0 = airTemp\_pi1 + 34.61\*airTemp\_err0 - 34.6\*airTemp\_err1

airTemp\_pi1 = airTemp\_pi0

await pwmFan(airTemp\_pi0)

}

await pwmFan(0)

**Ejecución del control PI de la temperatura del aire cada 100ms**

const airTemperature = new five.Thermometer({

controller: "LM35",

pin: "A5",

freq: 100

})

let airTemp1 = 0

airTemperature.on("data", function () {

let airTemp0 = this.celsius \* 0.0609 + airTemp1 \* 0.9391

airTempOut = airTemp0

airTemp1 = airTemp0

await airTempPidController(airTempSp)

})

**Control PI del nivel de la solución nutritiva**

const motor = new five.Motor({

pins: { dir: 8, pwm: 9 }, invertPWM: true

})

async function pwmPump (x) {

if (x < 200 || tnkLevel\_err0 < 0) {

await motor.stop(0)

} else if (x > 255) {

await motor.stop(0)

await motor.fwd(255)

} else {

await motor.stop(0)

await motor.fwd(x)

}

}

let tnkLevel\_pi0 = 0

let tnkLevel\_pi1 = 0

let tnkLevel\_err0 = 0

let tnkLevel\_err1 = 0

async function tnkLevelPidController (sp) {

tnkLevel\_err1 = tnkLevel\_err0

tnkLevel\_err0 = sp - tnkLevelOut

let tnkLevel\_pi0 = tnkLevel\_pi1 + 165.8 \* tnkLevel\_err0 - 165.7 \* nkLevel\_err1

tnkLevel\_pi1 = tnkLevel\_pi0

await pwmPump(tnkLevel\_pi0 \* 5)

}

await pwmPump(0)

**Ejecución del control PI del nivel de solución nutritiva cada 100ms**

const proximity = new five.Proximity({

controller: "HCSR04",

pin: 6,

freq: 100

})

let tnkLevel1 = 0

proximity.on("data", async function () {

let tnkLevel0 = this.cm \* 0.0609 + tnkLevel1 \* 0.9391

tnkLevelOut = 12.1 - tnkLevel0

tnkLevel1 = tnkLevel0

await tnkLevelPidController(tnkLevelSp)

})

De esta forma los setpoints indicados por el usuario se llegan a reflejar en el sistema de forma instantánea para los controles On – Off, por PWM y en un tiempo de establecimiento determinado para los controles PI.

Los controladores PI digitales implementados anteriormente fueron determinados en el software MathLab, en base a un conjunto de valores que fueron obtenidos por medio de la implementación de funciones diseñadas para ese fin.

FALTA!!!

## **Análisis de los valores almacenados para hallar valores optimizados para los parámetros ambientales.**

Entrar a la base de datos psql con Python, jalar los datos almacenados y la imagen de la webcam, luego aplicar el algoritmo genético.

## **Relación de los valores optimizados con los valores deseados.**

Del archivo de Python, generar un JSON. Ese JSON jalarlo desde el server.js del módulo web con process.env al presionar modo automático y yaaaaaaa!

# **CAPÍTULO IV – DISCUCIÓN**

# **CAPÍTULO V – CONCLUSIONES**

# **CAPÍTULO VI - SUGERENCIAS**

# **CAPÍTULO VII - BIBLIOGRAFÍA**

# **CAPÍTULO VIII – ANEXOS**