Sprint #3

Fecha de desarrollo: 24 de Abril - 29 de Abril

Equipo: Juan Andrés Abella Ballén (Scrum Máster), Santiago Cortes Tovar (Product Owner), Tomás Montañez Piñeros (Developer)

Product Backlog

Product Backlog		
	peso	horas
Construcción de la base	1	6
Construcción de la estructura del cabezal	2	12
Construcción de la zona donde se va a almacenar la electrónica	2	12
Colocar las guias y varillas roscadas por cada eje	2	12
Colocar los motores con cada varilla roscada	2	12
Poner la motobomba	1	6
Conectar los motores DC al puente H	1	6
Conectar el motor paso a paso al driver y mosfet a motobomba	1	6
Probar cada conexión	3	18
Conectar a la ESP32 con sensor humedad y colocarlo en la estructura	1	6
Programa de pruebas y probar sensor temperatura - humedad	2	12
Conectar a la ESP32 con microfono y colocarlo en la estructura	1	6
Programa de pruebas y probar microfono	2	12
Conectar a la ESP32 con camara y colocarlo en la estructura	2	12
Programa de pruebas y probar camara	4	24
Conectar las dos ESP32	1	6
Verificar la comunicación entre dispositivos	2	12
Implementar botones de control de parada de emergencia e incio	1	6
Tomar comandos de voz para entrenamiento	2	12
Entrenar la red neuronal para solución NLPE	4	24
Tomar diferentes posiciones del cabezal para entrenamiento	3	18
Entrenar la red neuronal de camara	5	30
Programar el movimiento del cabezal a lugar deseado	4	24
Programar el accionamiento de la motobomba	2	12
Reconocer a traves de la camara el estado de las plantas	4	24
Programar mensajes de alarma de plantas	2	12
Probar el movimiento y accion sincronizada entre motores y motobomba	3	18
Probar mensajes de emergencia	1	6
Construir conexión de red local	2	12
Hacer pruebas de conexión de red	3	18
Diseñar HMI y construirlo en código	2	12
Montaje de código en red local	1	6
Implementar conexión de datos a la red	3	18
Implementar conexión de recepción en la red	4	24
	76	

Para la realización del proyecto se realizó el anterior producto backlog, el cual tiene 34 actividades. A cada actividad se le asigno un peso de 1 hasta 5 dependiendo del grado de dificultad de la actividad, donde 1 corresponde a una tarea muy fácil, 2 a fácil, 3 a intermedió, 4 a difícil y por último 5 a muy difícil. Haciendo la sumatoria de todos los pesos asignados, se tiene que el peso de todo el proyecto es de 76.

Para la asignación de las horas demandadas por cada tarea se tuvo en cuenta las horas disponibles promedio de los integrantes del grupo, el cual es de 6 horas diarias. Por lo tanto, cada integrante puede aportar al proyecto 30 horas semanales. Considerando que

una tarea muy difícil requiere de mayor tiempo que una tarea fácil, se realizaron los siguientes intervalos de tiempo según su dificultad.

1. Muy fácil: (0 horas-6 Horas)
2. Fácil: (7 horas-12 horas)
3. Intermedio (13 horas – 18 horas)
4. Dificil (19 horas - 24 horas)
5. Muy Dificil (25 horas – 30 horas)

Sprint Planning

Se cuenta con el objetivo general de este sprint el cual es:

- Entrenamiento de red neuronal para identificación de acciones por comandos de voz y programación de recorrido mediante entrenamiento de inteligencia artificial para la cámara implementada.

El criterio de éxito para el sprint #3 es el entrenamiento del modelo adecuado tanto para los comandos de voz, como para el procesamiento de imágenes para el movimiento de la CNC.

Descripción detallada del product Backlog para el Sprint #3 y avances Sprint #2 y finalización del Sprint #1:

• Tomar comandos de voz para entrenamiento: Convertir voz a texto mediante el micrófono para poder procesar esta información con el microcontrolador y ejecutar acciones como regar plantas, plantar, extraer maleza o diagnosticar.

• Entrenar la red neuronal para solución NLPE: Obtener el modelo (Pesos y Sesgos) para los comandos extraídos por voz garantizando que el comportamiento de dicho modelo sea el adecuado.

• Tomar diferentes posiciones del cabezal para entrenamiento: Mediante el procesamiento de imágenes se debe poner indicar el recorrido del cabezal de la CNC, permitiendo el desplazamiento adecuado en los ejes X, Y e Z.

• Entrenar la red neuronal de cámara: Obtener el modelo (Pesos y Sesgos) del uso de la cámara para garantizar el movimiento adecuado de la CNC para que ejecute sus respectivas tareas de forma ordenada.

• Programar el movimiento del cabezal a lugar deseado: Ajuste del movimiento del cabezal al lugar deseado mediante criterios de escalamiento de imágenes para medición de posición.

• Programar el accionamiento de la motobomba: Mediante el programa verificar el accionamiento de la bomba de agua ajustando el tiempo de accionamiento para un riego ideal.

• Reconocer a través de la cámara el estado de las plantas: Diagnosticar la necesidad de las plantas ya sea si es necesario verificar el terreno para plantar, extracción de maleza, y riego periódico de las mismas.

• Programar mensajes de alarma de plantas: Los mensajes de alarma servirán para diagnosticar la planta si esta marchita o si es necesario que el agricultor realice algún proceso adicional de cuidado.

• Probar el movimiento y acción sincronizada entre motores y motobomba: Se debe garantizar que el riego en la zona de plantado sea uniforme para mantener condiciones estables para las plantas.

• Probar mensajes de emergencia: Identificar si hay algún comportamiento anormal que sea necesario activar mensajes de emergencia. (Por ejemplo, cabezal atascado, fallo en el riego, entre otros).

Teniendo en cuenta lo anterior se tiene el siguiente sprint backlog con la repartición de tareas para los integrantes.

Sprint 3						
Objetivo	Entrenamiento de red neuronal y programacion de recorrido			PLANNING		
Sprint Backlog		Peso	horas	**Santiago**	**Tomas**	**Juan**
1	Tomar comandos de voz para entrenamiento	2	12			12
2	Entrenar la red neuronal para solución NLPE	4	24		6	6
3	Tomar diferentes posiciones del cabezal para entrenamiento	3	18	3	10	3
4	Entrenar la red neuronal	5	30	10	10	10
5	Programar el movimiento del cabezal a lugar deseado	4	24	8	8	8
6	Programar el accionamiento de la motobomba	2	12	4	4	4
7	Reconocer a traves de la camara el estado de las plantas	4	24	4	10	10
8	Programar mensajes de alarma de plantas	2	12	12		
9	Probar el movimiento y accion sincronizada entre motores y motobomba	3	18	6	6	6
10	Probar mensajes de emergencia	1	6	2	2	2
TOTAL SEMANA 3		30	180	49	56	61

Monitoreo y control

AVANCE DE SPRINT 1

Para dar continuidad al desarrollo del proyecto, en este sprint se retomaron las actividades que habían quedado incompletas del Sprint 1. Como se muestra a continuación:

9 de abril / 12 de abril – 19 de abril / 22 de abril

Sprint 1											
Objetivo	Construir la estructura física y conectar los componentes principales del CNC			PLANNING			MONITORING				
	Sprint Backlog	Peso	horas	Santiago	Tomas	Juan	Santiago	Tomas	Juan		
1	Construcción de la base	1	6	3	3		3	3			
2	Construcción de la estructura del cabezal	2	12			12			12		
3	Construcción de la zona donde se va a almacenar la electrónica	2	12	12			12				
4	Colocar las guias y varillas roscadas por cada eje	2	12		12			4			
5	Colocar los motores con cada varilla roscada	2	12	6		6	6		2		
6	Poner la motobomba	1	6			6			2		
7	Conectar los motores DC al puente H	1	6	6			6				
8	Conectar el motor paso a paso al driver y mosfet a motobomba	1	6		6		1	3	1		
9	Probar cada conexión	3	18	6	6	6	4	4	4		
TOTAL SEMANA 1		15	90	33	27	30	32	14	21	67	11.1666667
							96.97%	51.85%	70.00%		

Gracias a este trabajo adicional, se logró **completar al 100% la construcción de la estructura de la CNC cuidadora de plantas**, avanzando desde un 74% de progreso registrado anteriormente.

(24 de Abril - 29 de Abril)

Sprint 1											
Objetivo	Construir la estructura física y conectar los componentes principales del CNC			PLANNING			MONITORING				
	Sprint Backlog	Peso	horas	Santiago	Tomas	Juan	Santiago	Tomas	Juan		
1	Construcción de la base	1	6	3	3		3	3			
2	Construcción de la estructura del cabezal	2	12			12			12		
3	Construcción de la zona donde se va a almacenar la electrónica	2	12	12			12				
4	Colocar las guias y varillas roscadas por cada eje	2	12		12			12			
5	Colocar los motores con cada varilla roscada	2	12	6		6	6	1	5		
6	Poner la motobomba	1	6			6		2	4		
7	Conectar los motores DC al puente H	1	6	6			6				
8	Conectar el motor paso a paso al driver y mosfet a motobomba	1	6		6		1	3	2		
9	Probar cada conexión	3	18	6	6	6	6	6	6		
TOTAL SEMANA 1		15	90	33	27	30	34	27	29	90	15
							103.03%	100.00%	96.67%		

Con trabajos individuales de 103.03%, 100% y 96.67% de Santiago, Tomas y Juan respectivamente.

AVANCE DE SPRINT 2

Una vez finalizadas las tareas pendientes del Sprint 1, el equipo procedió a avanzar con las actividades del Sprint 2, que habían quedado parcialmente desarrolladas. Para ello, se retomó el **Sprint Backlog correspondiente**, tal como se muestra a continuación:

9 de abril / 12 de abril – 19 de abril / 22 de abril

Sprint 2											
Objetivo	Implementacion de sensores y demas para el control del sistema			PLANNING			MONITORING				
	Sprint Backlog	Peso	horas	Santiago	Tomas	Juan	Santiago	Tomas	Juan		
1	Conectar a la ESP32 con sensor humedad y colocarlo en la estructura	1	6	6			4				
2	Programa de pruebas y probar sensor temperatura - humedad	2	12	12			12				
3	Conectar a la ESP32 con microfono y colocarlo en la estructura	1	6			6			4		
4	Programa de pruebas y probar microfono	2	12			12			12		
5	Conectar a la ESP32 con camara y colocarlo en la estructura	2	12		12			10			
6	Programa de pruebas y probar camara	4	24	6	12	6					
7	Conectar las dos ESP32	1	6	2	2	2	2	2	2		
8	Verificar la comunicación entre dispositivos	2	12	3	6	3	2	4	2		
9	Implementar botones de control de parada de emergencia e incio	1	6	2		2	2		2		
TOTAL SEMANA 2		16	96	31	32	31	22	16	22	60	10
							70.97%	50.00%	70.97%		

Durante esta etapa, se logró aumentar el progreso del sprint del **62.5% al 83.3%**, con lo cual se consolidaron gran parte de las tareas técnicas relacionadas con sensores y comunicación. Aún quedan pendientes la **integración funcional de la cámara** y la **verificación completa de conexión entre todos los componentes**.

(24 de Abril - 29 de Abril)

Sprint 2											
Objetivo	Implementacion de sensores y demas para el control del sistema			PLANNING			MONITORING				
	Sprint Backlog	Peso	horas	Santiago	Tomas	Juan	Santiago	Tomas	Juan		
1	Conectar a la ESP32 con sensor humedad y colocarlo en la estructura	1	6	6			6				
2	Programa de pruebas y probar sensor temperatura - humedad	2	12	12			12				
3	Conectar a la ESP32 con microfono y colocarlo en la estructura	1	6			6			6		
4	Programa de pruebas y probar microfono	2	12			12			12		
5	Conectar a la ESP32 con camara y colocarlo en la estructura	2	12		12			12			
6	Programa de pruebas y probar camara	4	24	6	12	6	2	10	2		
7	Conectar las dos ESP32	1	6	2	2	2	2	2	2		
8	Verificar la comunicación entre dispositivos	2	12	3	6	3	2	4	2		
9	Implementar botones de control de parada de emergencia e inicio	1	6	2		2	2		2		
TOTAL SEMANA 2		16	96	31	32	31	26	28	26	80	13.3333333
							83.87%	87.50%	83.87%		

Con respecto al 2 sprint se tiene un trabajo individual de 83.87%, 87.5% y 83.87% de Santiago, Tomas y Juan respectivamente.

DESARROLLO SPRINT 3

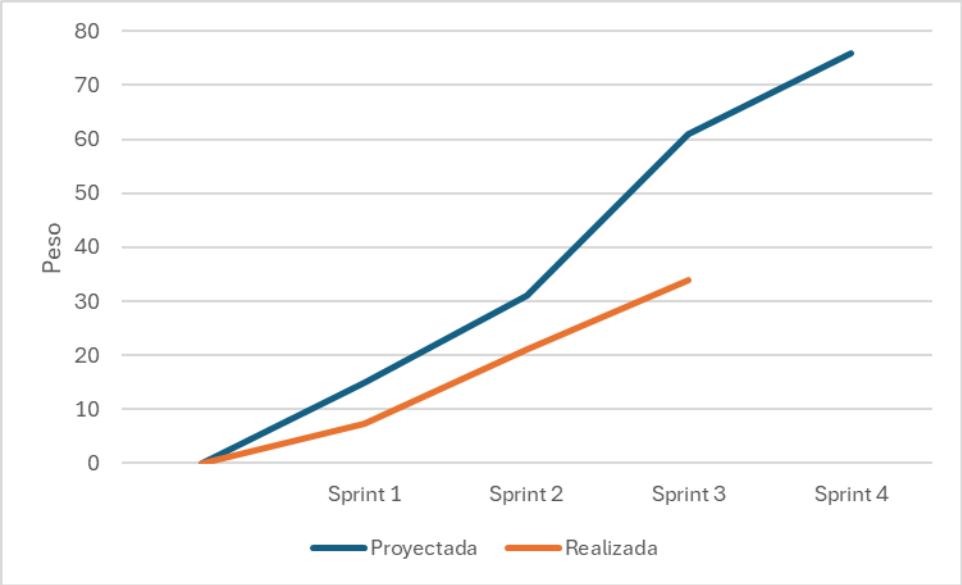
Durante el Sprint 3, el objetivo principal fue **el entrenamiento de la red neuronal y la programación de los movimientos de recorrido del sistema.** Debido a la gran complejidad de estas tareas no se pudo completar. Sin embargo, ya se tienen todas las herramientas necesarias para realizar el entrenamiento de la red neuronal, como lo es capturar la señal de audio del micrófono y capturar imágenes en la cámara. Es necesario tener muchos datos de entrenamiento para crear un modelo robusto que no sea sensible a perturbaciones.

(24 de Abril - 29 de Abril)

Sprint 3											
Objetivo	Entrenamiento de red neuronal y programacion de recorrido			PLANNING			MONITORING				
	Sprint Backlog	Peso	horas	Santiago	Tomas	Juan	Santiago	Tomas	Juan		
1	Tomar comandos de voz para entrenamiento	2	12			12			4		
2	Entrenar la red neuronal para solución NLPE	4	24		6	6		0	3		
3	Tomar diferentes posiciones del cabezal para entrenamiento	3	18	3	10	3	0	5	0		
4	Entrenar la red neuronal de camara	5	30	10	10	10	0	0	0		
5	Programar el movimiento del cabezal a lugar deseado	4	24	8	8	8					
6	Programar el accionamiento de la motobomba	2	12	4	4	4	4	4	4		
7	Reconocer a traves de la camara el estado de las plantas	4	24	4	10	10	1	3	2		
8	Programar mensajes de alarma de plantas	2	12	12			4				
9	Probar el movimiento y accion sincronizada entre motores y motobomba	3	18	6	6	6	0	0	0		
10	Probar mensajes de emergencia	1	6	2	2	2	1	1	1		
TOTAL SEMANA 3		30	180	49	56	61	9	12	13	34	5.66666667
							18.37%	21.43%	21.31%		

Burn up

En el desarrollo de este sprint se aumento en un 16.8% el progreso del proyecto, teniendo un resultado global del proyecto realizado de 34 de peso lo que representa un 44.7% completo. Teniendo en cuenta lo anterior se tiene la siguiente curva de burn up.



El presente documento resume el desarrollo y avance correspondiente al Sprint 3 del proyecto CNC, incluyendo tareas completadas, decisiones técnicas y distribución del trabajo entre los integrantes. Se entrega este informe como constancia del progreso alcanzado hasta la fecha, quedando a la espera de observaciones y recomendaciones por parte del responsable del seguimiento.

A continuación, se firma por parte del equipo desarrollador y el evaluador del proyecto como evidencia de revisión y validación:

Firma de Scrum Master

_____

Firma de Product Owner

Santiago Cortes Tovar

_____

Firma de Developer

_____

Firma de Cliente

_____

ANEXOS

- *Código de Cámara*

```
#include "OV7670.h"
#include <WiFi.h>
#include <WiFiMulti.h>
#include <WiFiClient.h>
#include "BMP.h"

const int SIOD = 21; // SDA
const int SIOC = 22; // SCL

const int VSYNC = 34;
const int HREF = 35;

const int XCLK = 32;
const int PCLK = 33;

const int D0 = 27;
const int D1 = 17;
const int D2 = 16;
const int D3 = 15;
const int D4 = 14;
const int D5 = 13;
const int D6 = 12;
const int D7 = 4;

#define ssid1    "Familia montañez"
#define password1 "Tomashilda243"

OV7670 *camera;
WiFiMulti wifiMulti;
WiFiServer server(80);

unsigned char bmpHeader[BMP::headerSize];

void serve()
{
 WiFiClient client = server.available();
 if (!client) return;

 String req = client.readStringUntil('\r');
 client.read(); // consume '\n'

 if (req.indexOf("GET /camera") >= 0) {
  camera->oneFrame();
  client.println("HTTP/1.1 200 OK");
  client.println("Content-type: image/bmp");
  client.println("Connection: close");
  client.println();
  client.write(bmpHeader, BMP::headerSize);
  client.write(camera->frame, camera->xres * camera->yres *
2);
 } else {
  client.println("HTTP/1.1 200 OK");
  client.println("Content-type: text/html");
  client.println("Connection: close");
  client.println();
  client.print(
   "<style>body{margin:0}
img{height:100%;width:auto}</style>"
   "<img id='a' src='/camera'
onload='this.style.display=\"initial\"; var
b=document.getElementById(\"b\"); b.style.display=\"none\";
b.src=\"/camera?\"+Date.now();'>"
```

```
    "<img id='b' style='display:none' src='/camera' onload='this.style.display=\"initial\"; var a=document.getElementById(\"a\"); a.style.display=\"none\"; a.src=\"/camera?\"+Date.now();'>"
   );
 }

 delay(1);
 client.stop();
}

void setup()
{
 Serial.begin(115200);

 wifiMulti.addAP(ssid1, password1);
 Serial.println("Connecting Wifi...");
 while (wifiMulti.run() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
 }

 Serial.println("\nWiFi connected");
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());

 camera = new OV7670(OV7670::Mode::QQVGA_RGB565, SIOD, SIOC, VSYNC, HREF, XCLK, PCLK, D0, D1, D2, D3, D4, D5, D6, D7);
 BMP::construct16BitHeader(bmpHeader, camera->xres, camera->yres);

 server.begin();
}

void loop()
{
 serve();
 camera->oneFrame();
}
```

Imagen obtenida de cámara OV7670 https://github.com/juanandres003/Proyecto-DIIN/tree/main/simulaciones/ESP32_I2S_Camera/ESP32_I2S_Camera_git

- Codigo extracción de señal de audio para entrenamiento para solución NLPE

```
#include <driver/i2s.h>

// Parámetros de I2S
#define I2S_WS  25
#define I2S_SD  32
#define I2S_SCK 33

#define SAMPLE_RATE 16000
#define SAMPLE_BITS I2S_BITS_PER_SAMPLE_16BIT
#define CHANNEL_FORMAT
I2S_CHANNEL_FMT_ONLY_LEFT
#define I2S_PORT I2S_NUM_0

// Configuración del buffer de energía
#define WINDOW_MS
10                          // Cada cuánto
calcular la energía (milisegundos)
```

```
#define VECTOR_DURATION_SEC
2               // Duración de la grabación
(segundos)
#define VECTOR_SIZE (VECTOR_DURATION_SEC *
1000 / WINDOW_MS) // Tamaño del vector de
energía

// Umbral para detectar voz
#define ENERGY_THRESHOLD 150

// Variables
float energyVector[VECTOR_SIZE];
bool recording = false;
unsigned long recordingStartTime = 0;
int vectorIndex = 0;

void setupI2S() {
  const i2s_config_t i2s_config = {
```

```
        .mode = i2s_mode_t(I2S_MODE_MASTER | I2S_MODE_RX),
        .sample_rate = SAMPLE_RATE,
        .bits_per_sample = SAMPLE_BITS,
        .channel_format = CHANNEL_FORMAT,
        .communication_format = I2S_COMM_FORMAT_I2S,
        .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1,
        .dma_buf_count = 8,
        .dma_buf_len = 512,
        .use_apll = false,
        .tx_desc_auto_clear = false,
        .fixed_mclk = 0
      };

      const i2s_pin_config_t pin_config = {
        .bck_io_num = I2S_SCK,
        .ws_io_num = I2S_WS,
        .data_out_num = I2S_PIN_NO_CHANGE,
        .data_in_num = I2S_SD
      };

      i2s_driver_install(I2S_PORT, &i2s_config, 0, NULL);
      i2s_set_pin(I2S_PORT, &pin_config);
      i2s_zero_dma_buffer(I2S_PORT);
    }

    float calculateEnergy() {
      const int samplesToRead = SAMPLE_RATE * WINDOW_MS / 1000;
      int16_t buffer[samplesToRead];
      size_t bytes_read;

      i2s_read(I2S_PORT, (void*)buffer, sizeof(buffer), &bytes_read, portMAX_DELAY);

      int samples_read = bytes_read / sizeof(int16_t);
      uint32_t sum = 0;

      for (int i = 0; i < samples_read; i++) {
        sum += abs(buffer[i]);
      }
```

```
      return (float)sum / samples_read;
    }

    void printEnergyVector() {
      Serial.println("\nEnergía grabada durante 2 segundos:");
      for (int i = 0; i < VECTOR_SIZE; i++) {
        Serial.print(energyVector[i]);
        Serial.print(",");
      }
      Serial.println("\nGrabación completa.");
      Serial.println("----------------------------");
    }

    void setup() {
      Serial.begin(115200);
      setupI2S();
    }

    void loop() {
      float currentEnergy = calculateEnergy();

      // SIEMPRE imprimir la energía actual
      Serial.println(currentEnergy);

      if (!recording) {
        // No estamos grabando aún
        if (currentEnergy > ENERGY_THRESHOLD) {
          // Se detectó voz, iniciar grabación
          Serial.println("¡Voz detectada! Iniciando grabación de 3 segundos...");
          recording = true;
          recordingStartTime = millis();
          vectorIndex = 0;
        }
      } else {
        // Estamos grabando
        if (vectorIndex < VECTOR_SIZE) {
          energyVector[vectorIndex] = currentEnergy;
          vectorIndex++;
        }
```

```
        if (millis() - recordingStartTime >= VECTOR_DURATION_SEC * 1000) {
          // Terminar grabación
          recording = false;
          printEnergyVector();
        }
      }

      delay(WINDOW_MS);
    }
```