

Sprint #2

Fecha de desarrollo: 9 de abril / 12 de abril – 19 de abril / 22 de abril

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

Para este sprint no se realizaron modificaciones en el product backlog, por lo tanto, se cuenta con el siguiente product backlog:

Product Backlog		
	Peso	Dias
Construcción de la estructura de la CNC	1	1
Ensamblaje y acoplamiento de los motores, motobomba y guias en la estructura	1	1
Implementar el sistema de control de los actuadores del sistema	2	2
Implementación del electroiman	1	1
Implementación de sensor de humedad	1	1
Implementación de la camara	2	2
Implementacion de Solucion NLPE	5	7
Implementación del microfono	1	1
Entrenamiento de la Inteligencia Artificial	5	8
Implementar protocolo de pruebas	3	4
Comunicación entre microcontroladores y demás electrónica	2	2
Generación de red local con dirección IP	2	2
Programar riego por zonas	4	5
Programar sistema de monitoreo del estado de las plantas	4	5
Diseñar HMI y de alertas	1	1
Enviar datos y recibir datos desde conexión IP	2	2
	37	45

Para el entregable de este sprint se planeó el siguiente sprint backlog:

Sprint 2				PLANNING			
Objetivo	Implementacion de sensores y demas para el control del sistema	Peso	Dias		Santiago	Tomas	Juan
Sprint Backlog							
1	Implementación del electroiman	1	1	Conectar el electroiman a su actuador y colocarlo en el cabezal	0,1	0,1	
				Programar un programa de pruebas y probar	0,9	0,9	
2	Implementación de sensor de humedad	1	1	Conectar a la ESP32 y colocarlo en la estructura	0,1		0,1
				Programar un programa de pruebas y probar	0,9		0,9
3	Implementación del microfono	1	1	Conectar a la ESP32 y colocarlo en la estructura	0,1		0,1
				Programar un programa de pruebas y probar	0,9		0,9
4	Implementación de la camara	2	2	Conectar a la ESP32 y colocarlo en la estructura	0,1		0,1
				Programar un programa de pruebas y probar	1,9	0,64	0,64
5	Implementación de la camara	2	2	Conectar las dos ESP32	0,1		0,1
				Verificar la comunicación	0,4	0,2	0,2
				Implementar botones de control de parada de emergencia e inicio	1,5	0,49	0,49
TOTAL SEMANA 2					7	2,33	2,33
							2,34

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

Ajuste e implementación sensores y de componentes necesarios para el movimiento de la estructura acoplando diferentes piezas diseñadas para permitir los movimientos en el eje X, Y e Z contando con la puesta en funcionamiento de los tres motores.

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

- ## AVANCE DE SPRINT 1

Antes de dar inicio al Sprint 2, el equipo continuó trabajando en el diseño y ensamblaje del montaje físico de la estructura CNC, lo que implicó ajustar algunas de las tareas originalmente planificadas. Como se muestra en la tabla, se reorganizó el trabajo para enfocarse en completar los elementos pendientes del Sprint 1:

Sprint 1				(3 de Abril - 8 de Abril)											
Objetivo	Construir la estructura física y conectar los componentes principales del CNC	Peso	Días	PLANNING						MONITORING					
Sprint Backlog				Santiago	Tomas	Juan	Santiago	Tomas	Juan	Santiago	Tomas	Juan			
1	Construcción de la estructura de la CNC	1	1	Construcción de la base	0.4	0.2	0.2		0.2	0.2					
				Construcción de la estructura del cabezal	0.4			0.4			0.4				
				Construcción de la zona donde se va a almacenar la electrónica	0.2	0.2		0.1							
				Colocar las guías y varillas roscadas por cada eje	0.5		0.5		0						
2	Ensamblaje y acoplamiento de los motores, motobomba y guías en la estructura	1	1	Colocar los motores con cada varilla roscada	0.4	0.2		0.2	0.2		0				
				Poner la motobomba	0.1		0.1			0					
				Conectar los motores DC al puente H	0.1	0.1		0.1							
3	Implementar el sistema de control de los actuadores del sistema	2	2	Conectar el motor paso a paso al driver	0.1		0.1		0.1						
				Conectar la motobomba al mosfet	0.1			0.1			0				
				Probar cada conexión	1.7	0.6	0.55	0.55	0.1	0.5	0.5				
					4	1.3	1.35	1.35	0.7	0.8	0.9	2.4			
TOTAL SEMANA 1				4	4					53.85%	59.76%	66.67%			

Se evidencian avances en el montaje estructural, con distribución del trabajo entre los tres integrantes del equipo:

Sprint 1				(9 de Abril / 12 de Abril - 19 de Abril / 22 de Abril)											
Objetivo		Construir la estructura física y conectar los componentes principales del CNC				PLANNING						MONITORING			
Sprint Backlog				Peso	Días	Santiago	Tomas	Juan	Santiago	Tomas	Juan				
1	Construcción de la estructura de la CNC	1	1	Construcción de la base	0,4	0,2	0,2		0,2	0,2					
				Construcción de la estructura del cabezal	0,4			0,4			0,4				
				Construcción de la zona donde se va a almacenar la electrónica	0,2	0,2			0,2						
2	Ensamblaje y acoplamiento de los motores, motobomba y guías en la estructura	1	1	Colocar las guías y varillas roscadas por cada eje	0,5		0,5			0,2					
				Colocar los motores con cada varilla roscada	0,4	0,2		0,2	0,2		0				
				Poner la motobomba	0,1			0,1			0,05				
3	Implementar el sistema de control de los actuadores del sistema	2	2	Conectar los motores DC al puente H	0,1	0,1			0,1						
				Conectar el motor paso a paso al driver	0,1		0,1			0,1					
				Conectar la motobomba al mosfet	0,1			0,1			0				
				Probar cada conexión	1,7	0,6	0,55	0,55	0,3	0,55	0,55				
TOTAL SEMANA 1				4	4	4	1,3	1,35	1,35	1	1,05	1	3,05		
										76,92%	77,78%	74,07%			

Para lo cual se puede expresar la finalización de la construcción de la estructura de la CNC, completando en su mayoría el ensamblaje y acoplamiento de los motores, motobomba, guías y los diferentes dispositivos en la maqueta física y con ello haciendo que cada uno de los integrantes complete satisfactoriamente una parte de sus tiempos y pesos de trabajo.

Con ello se tiene que se pasó de un desarrollo del 60% al 76.25% y para ser completado solo se debe completar el montaje de la CNC.

DESARROLLO SPRINT 2

Durante este segundo sprint, el enfoque principal estuvo en la **implementación, conexión y pruebas de los dispositivos electrónicos** que forman parte del sistema CNC. Según lo planteado en el Sprint Backlog, las tareas se centraron en validar la comunicación entre los microcontroladores y los distintos sensores que se integrarán al diseño.

En esta etapa se trabajó en:

- La conexión y prueba del **sensor de humedad y temperatura**.
- La verificación del **funcionamiento del micrófono** como entrada de datos.
- La **integración de la cámara**, considerando calidad de captura y compatibilidad.
- La **comunicación entre los microcontroladores** y cada uno de los dispositivos mencionados.

Estas actividades permitieron avanzar significativamente en la preparación electrónica del sistema, sentando las bases para su integración final con la estructura física. Los resultados de este sprint se resumen en la siguiente tabla:

Sprint 2				(9 de Abril / 12 de Abril - 19 de Abril / 22 de Abril)									
Objetivo	Implementación de sensores y demas para el control del sistema				PLANNING			MONITORING					
	Sprint Backlog	Peso	Dias		Santiago	Tomas	Juan	Santiago	Tomas	Juan			
1	Implementación del electroiman	1	1	Conectar el electroiman a su switch y colocarlo en el cableado Programar un programa de pruebas y probar	0.1	0.1							
					0.9	0.9							
2	Implementación de sensor de humedad	1	1	Conectar a la ESP32 y colocarlo en la estructura Programar un programa de pruebas y probar	0.1		0.1		0.05				
					0.9		0.9		0.6		0.3		
3	Implementación del microfono	1	1	Conectar a la ESP32 y colocarlo en la estructura Programar un programa de pruebas y probar	0.1			0.1	0.2	0.1	0.05		
					0.9			0.9			0.6		
4	Implementación de la camara	2	2	Conectar a la ESP32 y colocarlo en la estructura Programar un programa de pruebas y probar	0.1			0.1	0.05				
					1.9	0.64	0.64	0.62	0.2	0.2	0.2		
5	Comunicación entre microcontroladores y demás electrónica	2	2	Conectar las dos ESP32 Verificar la comunicación	0.1			0.1	0.1				
					0.4	0.2	0.2			0.3			
				Implementar botones de control de parada de emergencia e inicio	1.5	0.49	0.49	0.52	0.4	0.6	0.5		
TOTAL SEMANA 2		7	7		7	2.33	2.33	2.34	1.6	1.2	1.68	4.45	
								68.67%			51.50%		
											70.51%		

Firma de Scrum Master



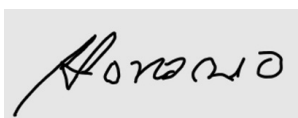
Firma de Product Owner

Santiago Cortes Tovar

Firma de Developer



Firma de Cliente



Anexos

Programa de ESP32 para control de motores y recolección de datos del DHT11

```
#include "DHT.h"

const int INTERVALO = 30; // Intervalo en ms para el
cambio de velocidad

#define DHTPIN 5 // Pin donde está conectado el sensor
DHT

#define DHTTYPE DHT22 // Tipo de sensor DHT (DHT11,
DHT22, etc.)

DHT dht(DHTPIN, DHTTYPE);

// Definición de pines para el L298N

const int motorPin1 = 6; // Pin PWM para controlar
velocidad en una dirección

const int motorPin2 = 7; // Pin PWM para controlar
velocidad en dirección opuesta

// Definición de pines para el L298N

const int motorPin3 = 8; // Pin PWM para controlar
velocidad en una dirección

const int motorPin4 = 9; // Pin PWM para controlar
velocidad en dirección opuesta

// Definición de pines para step motor

const int bob1 = 10;

const int bob2 = 11;

const int bob3 = 12;

const int bob4 = 13;

const int mis = 2.5; // 5 10 15

// Variables para el control del motor

int velocidad1 = 0, velocidad2 = 0; // Valor PWM (0-255)

bool direccion1 = true, direccion2 = true; // true =
adelante, false = atrás

bool modoAutomatico = false;

char step = 'p';

// Variables para el modo automático

unsigned long tiempoAnterior = 0;

void setup() {

// Configuración motor 1

pinMode(motorPin1, OUTPUT);

pinMode(motorPin2, OUTPUT);

// Configuración motor 2

pinMode(motorPin3, OUTPUT);

pinMode(motorPin4, OUTPUT);

// Configuración step motor

pinMode(bob1, OUTPUT);

pinMode(bob2, OUTPUT);

pinMode(bob3, OUTPUT);

pinMode(bob4, OUTPUT);

// Configuración dht22

dht.begin();

// Inicializar motor detenido

digitalWrite(motorPin1, LOW);

digitalWrite(motorPin2, LOW);

digitalWrite(motorPin3, LOW);

digitalWrite(motorPin4, LOW);

// Iniciar comunicación serial

Serial.begin(115200);

Serial.println("Control de Motor L298N");

Serial.println("Comandos disponibles:");

Serial.println("A+velocidad - Giro en sentido horario (0-
255)");

Serial.println("B+velocidad - Giro en sentido antihorario
(0-255)");
```

```

    Serial.println("C+velocidad - Giro en sentido horario (0-255)");

    Serial.println("D+velocidad - Giro en sentido antihorario (0-255)");

    Serial.println("E - Giro en sentido horario (step motor)");

    Serial.println("F - Giro en sentido antihorario (step motor)");

    Serial.println("S - Detener motor");

    Serial.println("M - Activar/desactivar modo automático");
}

void loop() {

    // Leer comandos del serial

    if (Serial.available() > 0) {

        String comando = Serial.readStringUntil('\n'); // Leer hasta nueva línea

        procesarComando(comando);

    }

    // Control automático si está activo

    if (modoAutomatico) {

        modoVariacionAutomatica();

    }

    if (step == 'E' || step == 'e') {

        stepdireccion(bob1, bob2, bob3, bob4, mis);

        Serial.println("Step motor girando en sentido horario");

    } else if (step == 'F' || step == 'f') {

        stepdireccion(bob4, bob3, bob2, bob1, mis);

        Serial.println("Step motor girando en sentido antihorario");

    } else if (step == 'P' || step == 'p'){

        digitalWrite(bob1, LOW);

        digitalWrite(bob2, LOW);

        digitalWrite(bob3, LOW);

        digitalWrite(bob4, LOW);

        Serial.println("step motor detenido");

    } else {

        Serial.println("Comando no reconocido");

```

```

    }

    // Leer temperatura y humedad

    float h = dht.readHumidity();

    float t = dht.readTemperature();

    Serial.print("Humedad: ");

    Serial.print(h);

    Serial.print("%, Temperatura: ");

    Serial.print(t);

    Serial.println("°C");

}

void procesarComando(String comando) {

    comando.trim(); // Eliminar espacios en blanco

    if (comando.length() > 0) {

        char primerCaracter = comando.charAt(0);

        step = primerCaracter;

        // Detener modo automático si se recibe cualquier comando

        if (primerCaracter != 'M' && modoAutomatico) {

            modoAutomatico = false;

            Serial.println("Modo automático desactivado");

        }

        switch (primerCaracter) {

            case 'A':

            case 'a':

                // Giro en sentido horario

                if (comando.indexOf('+') != -1) { // devuelve la posición del primer '+' encontrado

                    velocidad1 = comando.substring(comando.indexOf('+') + 1).toInt(); // extraer velocidad

                    velocidad1 = constrain(velocidad1, 0, 255); // Limitar velocidad entre 0 y 255

                    moverMotor1(true, velocidad1); // mover motor hacia adelante

                    Serial.print("Motor 1 girando hacia adelante con velocidad: ");

                    Serial.println(velocidad1);

```

```

    }

    break;

    case 'B':

    case 'b':

        // Giro en sentido antihorario

        if (comando.indexOf('+') != -1) {

            velocidad1 =
comando.substring(comando.indexOf('+') + 1).toInt();

            velocidad1 = constrain(velocidad1, 0, 255);

            moverMotor1(false, velocidad1);

            Serial.print("Motor 1 girando hacia atrás con
velocidad: ");

            Serial.println(velocidad1);

        }

        break;

    case 'S':

    case 's':

        // Detener motor

        detenerMotor();

        Serial.println("Motores detenidos");

        break;

    case 'M':

    case 'm':

        // Activar/desactivar modo automático

        modoAutomatico = !modoAutomatico;

        if (modoAutomatico) {

            Serial.println("Modo automático activado");

            velocidad1 = 0; // Iniciar desde velocidad 0

            paso = 1;    // Iniciar incrementando

        } else {

            Serial.println("Modo automático desactivado");

            detenerMotor();

        }

        break;

```

```

    case 'C':

    case 'c':

        // Giro en sentido horario para motor 2

        if (comando.indexOf('+') != -1) {

            velocidad2 =
comando.substring(comando.indexOf('+') + 1).toInt();

            velocidad2 = constrain(velocidad2, 0, 255);

            moverMotor2(true, velocidad2); // mover motor hacia
adelante

            Serial.print("Motor 2 girando hacia adelante con
velocidad: ");

            Serial.println(velocidad2);

        }

        break;

    case 'D':

    case 'd':

        // Giro en sentido antihorario para motor 2

        if (comando.indexOf('+') != -1) {

            velocidad2 =
comando.substring(comando.indexOf('+') + 1).toInt();

            velocidad2 = constrain(velocidad2, 0, 255);

            moverMotor2(false, velocidad2);

            Serial.print("Motor 2 girando hacia atrás con
velocidad: ");

            Serial.println(velocidad2);

        }

        break;

    default:

        Serial.println("Comando no reconocido");

        break;

    }

    }

}

void moverMotor1(bool direc, int vel) {

    if (direc) {

```



```

// Dirección 1 (adelante)

analogWrite(motorPin1, vel);

analogWrite(motorPin2, 0);

} else {

// Dirección 2 (atrás)

analogWrite(motorPin1, 0);

analogWrite(motorPin2, vel);

}

```

```

direccion1 = direc;

velocidad1 = vel;

}

```

```

void moverMotor2(bool direc, int vel){

if (direc) {

// Dirección 1 (adelante)

analogWrite(motorPin3, vel);

analogWrite(motorPin4, 0);

} else {

// Dirección 2 (atrás)

analogWrite(motorPin3, 0);

analogWrite(motorPin4, vel);

}

}

```

```

direccion2 = direc;

velocidad2 = vel;

}

```

```

void detenerMotor() {

analogWrite(motorPin1, 0);

analogWrite(motorPin2, 0);

analogWrite(motorPin3, 0);

analogWrite(motorPin4, 0);

// Reiniciar variables

velocidad1 = 0;

velocidad2 = 0;

}

```

```

void modoVariacionAutomatica() {

unsigned long tiempoActual = millis(); // Obtener tiempo actual

```

```

// Cambiar velocidad cada INTERVALO milisegundos

if (tiempoActual - tiempoAnterior >= INTERVALO) {

tiempoAnterior = tiempoActual; // Actualizar tiempo anterior

```

```

// Cambiar dirección de incremento/decremento en los límites

```

```

if (velocidad1 >= 255) {

paso = -1; // Comenzar a decrementar

} else if (velocidad1 <= 0) {

paso = 1; // Comenzar a incrementar

// Cambiar dirección de giro cuando llega a 0

direccion1 = !direccion1;

}

```

```

// Actualizar velocidad

velocidad1 += paso;

velocidad1 = constrain(velocidad1, 0, 255);

```

```

// Aplicar al motor

moverMotor1(direccion1, velocidad1);

```

```

// Mostrar información cada 10 pasos para no saturar el puerto serial

```

```

if (velocidad1 % 10 == 0) {

Serial.print("Auto - Dirección: ");

Serial.print(direccion1 ? "Adelante" : "Atrás");

Serial.print(", Velocidad: ");

Serial.println(velocidad1);

}

```

```

}

}

```



```

        .data_in_num = I2S_SD
    };

    i2s_driver_install(I2S_NUM_0, &i2s_config, 0, NULL);
    i2s_set_pin(I2S_NUM_0, &pin_config);
    i2s_zero_dma_buffer(I2S_NUM_0);
}

```

```

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

    setupI2S();
    delay(500);
}

```

```

void loop() {
    int32_t sampleBuffer[64];

```

```

    size_t bytesRead;

```

```

    // Leer datos del micrófono

```

```

    i2s_read(I2S_NUM_0, (void*)sampleBuffer,
    sizeof(sampleBuffer), &bytesRead, portMAX_DELAY);

```

```

    int samples = bytesRead / sizeof(int32_t);

```

```

    for (int i = 0; i < samples; i++) {

```

```

        // Convertir a valores más pequeños para graficar
        fácilmente

```

```

        int value = sampleBuffer[i] >> 14; // Reduce resolución
        (ajusta si lo ves muy plano)

```

```

        Serial.println(value);      // Una sola línea para que el
        Serial Plotter funcione bien

```

```

    }

```

```

}

```

Codigo para ESP32 para implementación de cámara

```

// Non-FIFO OV7670 Video on TFT with ESP32

```

```

#define MODE QVGA // 320 X 240

```

```

// #define MODE QQVGA // 160 x 120

```

```

// #define MODE QCIF // 176 x 144 (crop)

```

```

// #define MODE QQCIF // 88 x 72 (crop)

```

```

#define COLOR RGB565

```

```

// #define COLOR YUV422

```

```

#define ROTATION 1 // 0~4

```

```

#include <Wire.h>

```

```

#include <SPI.h>

```

```

#include <OV7670.h>

```

```

#include <Adafruit_GFX.h>

```

```

#include <Adafruit_ILI9341.h>

```

```

// OV7670 pins

```

```

// SSCB_SDA(SIOD) -> 21(ESP32)

```

```

// SSCB_SCL(SIOC) -> 22(ESP32)

```

```

// RESET      -> 3.3V

```

```

// PWDN          -> GND

```

```

// HREF          -> NC

```

```

const camera_config_t cam_conf = {

```

```

    .D0 = 36,

```

```

    .D1 = 39,

```

```

    .D2 = 34,

```

```

    .D3 = 35,

```

```

    .D4 = 32,

```

```

    .D5 = 33,

```

```

    .D6 = 25,

```

```

    .D7 = 26,

```

```

    .XCLK = 27,

```

```

    .PCLK = 14,

```

```

    .VSYNC = 13,

```

```

    .xclk_freq_hz = 10000000, // XCLK 10MHz

```

```

    .ledc_timer = LEDC_TIMER_0,

```

```
.ledc_channel = LEDC_CHANNEL_0
```

```
};
```

```
// TFT pins
```

```
#define TFT_DC 16
```

```
#define TFT_CS 5
```

```
#define TFT_RST 17
```

```
#define TFT_MISO 19
```

```
#define TFT_MOSI 23
```

```
#define TFT_CLK 18
```

```
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC,  
TFT_RST);
```

```
OV7670 cam;
```

```
uint16_t *buf, w, h;
```

```
void setup() {
```

```
    setCpuFrequencyMhz(240);
```

```
    Serial.begin(9600);
```

```
    Wire.begin();
```

```
    Wire.setClock(400000);
```

```
    tft.begin();
```

```
    tft.setRotation(ROTATION);
```

```
    tft.fillScreen(0);
```

```
    esp_err_t err = cam.init(&cam_conf, MODE, COLOR);
```

```
    if (err != ESP_OK) Serial.println("cam.init ERROR");
```

```
    cam.setPCLK(2, DBLV_CLK_x4);
```

```
    cam.vflip(false);
```

```
Serial.printf("cam MID = %X\n\r", cam.getMID());
```

```
Serial.printf("cam PID = %X\n\r", cam.getPID());
```

```
switch (MODE) {
```

```
    case QVGA:
```

```
        w = 320;
```

```
        h = 240;
```

```
        break;
```

```
    case QQVGA:
```

```
        w = 160;
```

```
        h = 120;
```

```
        break;
```

```
    case QCIF:
```

```
        w = 176;
```

```
        h = 144;
```

```
        break;
```

```
    case QQCIF:
```

```
        w = 88;
```

```
        h = 72;
```

```
    }
```

```
}
```

```
void loop(void) {
```

```
    for (uint16_t y = 0; y < h; y++) {
```

```
        buf = cam.getLine(y + 1);
```

```
        tft.drawRGBBitmap(0, y, buf, w, 1);
```

```
    }
```

```
}
```