Sprint #4

Fecha de desarrollo: 30 de Abril – 6 de Mayo

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

El criterio de éxito para el sprint #4 es el ajuste de información y comunicación del usuario mediante la interfaz gráfica que permita controlar la CNC para el cuidado de las plantas.

Descripción detallada del product Backlog para el Sprint #4 y avances Sprint #3 y finalización del Sprint #1 y Sprint #2:

- Construir conexión de red local: Activar el módulo WiFi de la ESP32 para configurar un punto de acceso (Access Point) que permita generar un servidor local, facilitando la conectividad del sistema en zonas rurales sin necesidad de infraestructura de red externa. Esta red servirá como medio principal de comunicación entre el usuario y el sistema de automatización.
- Hacer pruebas de conexión de red: Realizar pruebas funcionales y de estabilidad para verificar que la red WiFi local creada por la ESP32 es robusta, presenta baja latencia y se mantiene estable en condiciones variables, asegurando una comunicación confiable para el monitoreo y control automatizado de las plantas.
- Diseñar el HMI y construirlo en código: Diseñar una interfaz hombre-máquina (HMI) intuitiva y funcional que contenga botones interactivos para controlar el sistema CNC. Esta interfaz debe incluir opciones para el manejo de motores, así como el encendido y apagado del sistema de riego, todo implementado en HTML, CSS y JavaScript embebido en el servidor web local de la ESP32.
- Montaje de código en red local: Integrar la interfaz gráfica desarrollada con el servidor local de la ESP32, asegurando que los eventos de la HMI (como pulsación de botones) se comuniquen correctamente con el microcontrolador a través de peticiones HTTP o WebSocket. Este montaje permitirá el control en tiempo real del sistema mediante la red local.
- Implementar conexión de datos a la red: Desarrollar e integrar la lógica de envío de datos desde la ESP32 hacia la interfaz web, garantizando que las variables del sistema (por ejemplo, estados de sensores, posición de motores o humedad del suelo) se transmitan correctamente mediante tramas de datos en formato JSON o similar.
- Implementar conexión de recepción de la red: Configurar la recepción de comandos desde la interfaz gráfica hacia la ESP32, interpretando correctamente las solicitudes de control enviadas por el usuario. Este proceso debe asegurar una respuesta eficiente

para ejecutar acciones como el movimiento de motores, el riego o cualquier otra operación automatizada.

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

Sprint 4						
Objetivo	Envio de informacion e interfaz de usuario			PLANNING		
Sprint Backlog		Peso	horas	**Santiago**	**Tomas**	**Juan**
1	Construir conexión de red local	2	12	10	1	1
2	Hacer pruebas de conexión de red	3	18	6	6	6
3	Diseñar HMI y construirlo en código	2	12	8		4
4	Montaje de código en red local	1	6	6		
5	Implementar conexión de datos a la red	3	18			18
6	Implementar conexión de recepción en la red	4	24		23	1
TOTAL SEMANA 4		15	90	30	30	30

Monitoreo y Control

FINALIZACIÓN SPRINT 1

Para el sprint 1 ya se había hecho desde el sprint anterior la finalización del mismo con la construcción completa del sistema con motores y conexiones.

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%		

AVANCE SPRINT 2

Una vez finalizado el sprint 3, el equipo procedió a avanzar con las actividades faltante del sprint 2, que habia quedado parcialmente desarrollado se retomo el **Sprint Backlog correspondiente**, tal como se muestra a comunicación:

(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 incio	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%		

Durante esta etapa, se logró aumentar el progreso del sprint del **83.3% al 100%**, con lo cual se consolidaron gran parte de las tareas técnicas relacionadas con las pruebas de cámara y verificación de comunicación.

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	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	3	6	3			
9	Implementar botones de control de parada de emergencia e incio	1	6	3		3	3		3			
TOTAL SEMANA 2		16	96	32	32	32	32	32	32	96	16	
							100,00%	100,00%	100,00%			

Con respecto al 2 sprint se tiene un trabajo individual completo de 100%, 100% y 100% de Santiago, Tomas y Juan respectivamente.

AVANCE SPRINT 3

(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	0	0	0		
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%		

A partir del sprint anterior sprint, mostrada anteriormente, donde solo se logró el programa de accionamiento de la motobomba se pudo tomar datos para el entrenamiento de la red neuronal que reconocerá los comandos de voz para el riego de plantas, estos datos de entrenamiento se tomaron del código de Arduino anexado en **Anexo 2**. Se colocaron los datos en colab, se entrenó la red neuronal y se predijo con una entrada diferente a las de entrenamiento y prueba, vea **Anexo 3**.

Por otra parte, también se probó la cámara con las diferentes con diferentes posiciones del cabezal, sin embargo, se tuvieron unos problemas con la toma de datos de entrenamiento de la inteligencia artificial, debido a que las imágenes de la cámara se tuvieron que colocar en un servidor local además de tener en cuenta que al no haber finalizado la construcción de la CNC para ese momento dificultó en gran medida el entrenamiento de la Inteligencia artificial específicamente para el funcionamiento de la CNC.

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			12		
2	Entrenar la red neuronal para solución NLPE	4	24		6	6		6	6		
3	Tomar diferentes posiciones del cabezal para entrenamiento	3	18	3	10	3	3	10	3		
4	Entrenar la red neuronal de camara	5	30	10	10	10	5	5	5		
5	Programar el movimiento del cabezal a lugar deseado	4	24	8	8	8	4	4	4		
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	4	10	10		
8	Programar mensajes de alarma de plantas	2	12	12			12				
9	Probar el movimiento y accion sincronizada entre motores y motobomba	3	18	6	6	6	6	6	6		
10	Probar mensajes de emergencia	1	6	2	2	2	2	2	2		
TOTAL SEMANA 3		30	180	49	56	61	40	47	52	139	23,1666667
							81,63%	83,93%	85,25%		

Teniendo en cuanta lo anterior se trabajaron 139 horas de las 180 horas predestinadas para este sprint, lo que refleja un proceso del 77.2%, del cual Santiago trabajo un 81.63%, Tomas un 83.93% y Juan un 85.25%.

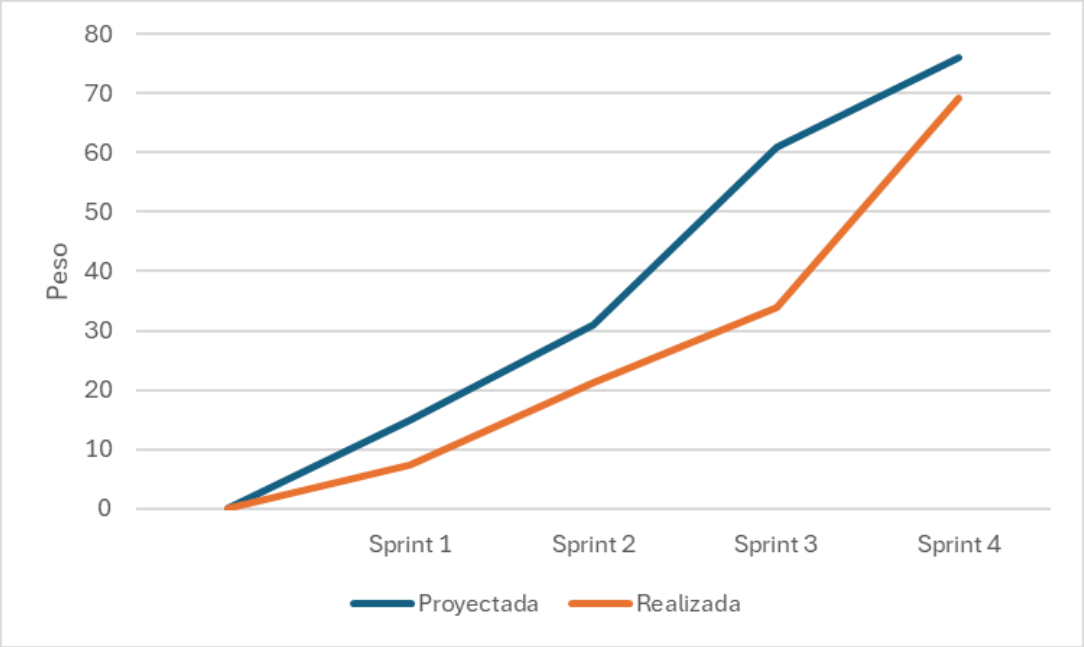
DESARROLLO SPRINT 4

Durante el Sprint 4, el objetivo principal fue establecer una **red local de comunicación** para el sistema, permitiendo así la **verificación y gestión remota de datos** desde la CNC cuidadora de plantas. Esta etapa se centró especialmente en el desarrollo de software, lo cual permitió avanzar con mayor agilidad en comparación con los sprints anteriores, ya que las tareas requeridas fueron más directas y menos dependientes del ensamblaje físico.

Sprint 4										
Objetivo	Envio de informacion e interfaz de usuario			PLANNING			MONITORING			
Sprint Backlog		Peso	horas	Santiago	Tomas	Juan	Santiago	Tomas	Juan	
1	Construir conexión de red local	2	12	10	1	1	10	1	1	
2	Hacer pruebas de conexión de red	3	18	6	6	6	6	6	6	
3	Diseñar HMI y construirlo en código	2	12	8		4	8		4	
4	Montaje de código en red local	1	6	6			6			
5	Implementar conexión de datos a la red	3	18			18			18	
6	Implementar conexión de recepción en la red	4	24		23	1		23	1	
TOTAL SEMANA 4		15	90	30	30	30	30	30	30	90
							100,00%	100,00%	100,00%	

Gracias a una adecuada distribución del trabajo y al enfoque en tareas de programación, este sprint alcanzó un **100% de cumplimiento** por parte de todos los integrantes del equipo: Santiago, Tomás y Juan, completando las 90 horas planificadas para esta semana de desarrollo.

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

Anexo 1: Fotos de la entrega al usuario

Anexo 2: Programa para sacar datos de entrenamiento

```
#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 20                // 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 250

// 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++;
    }
```

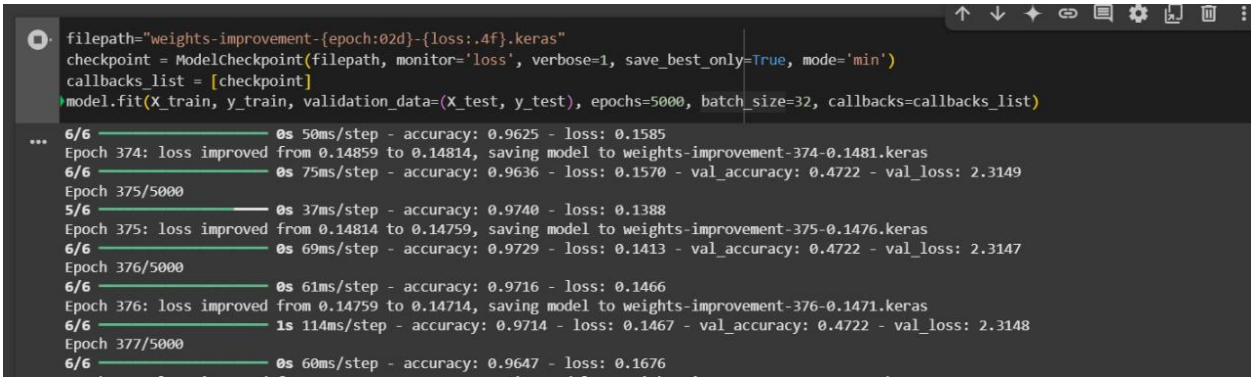
```
  if (millis() - recordingStartTime >=
VECTOR_DURATION_SEC * 1000) {
    // Terminar grabación
    recording = false;
    printEnergyVector();
  }
}

delay(WINDOW_MS);
}
```

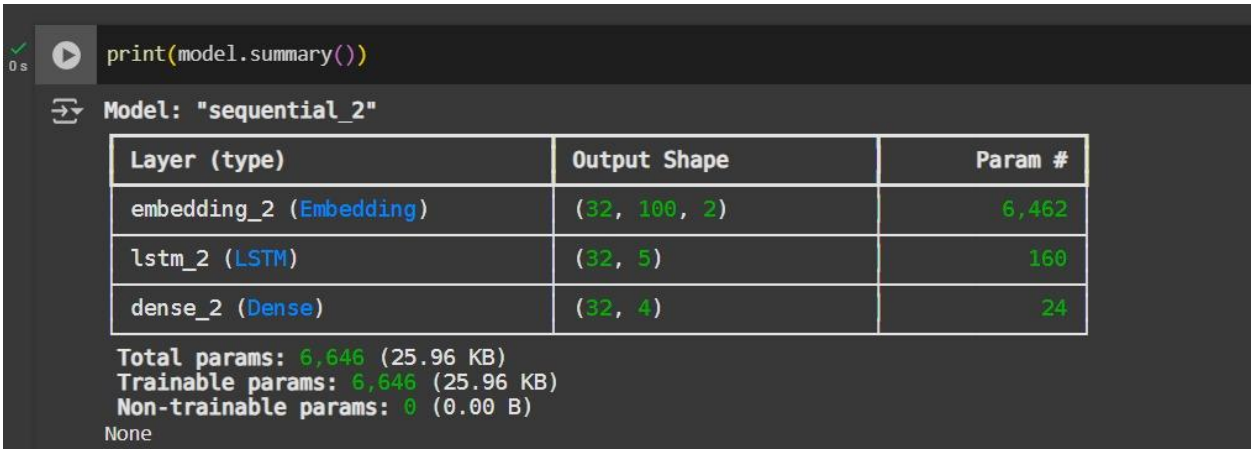
Anexo 3: Colab con datos de entrenamiento, entrenamiento y predicción con datos de voz

https://colab.research.google.com/drive/1AGqUdAUS_NVJa_aPPbYlB4TVAm7VHcJt?usp=sharing

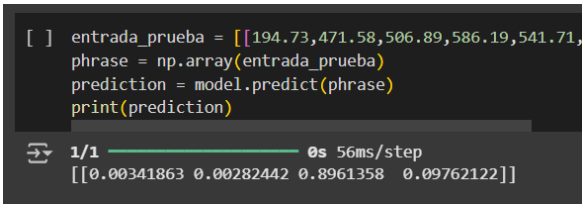
Entrenamiento

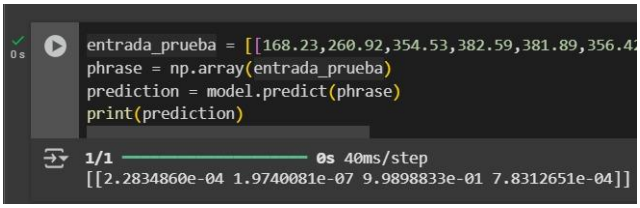


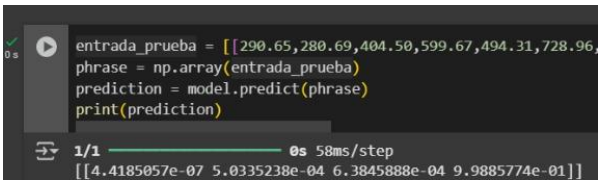
Peso de la red neuronal

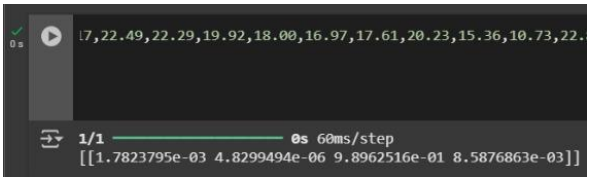


Predicciones









Anexo 4: Pagina web con servidor propio de la ESP32