



Universidad ORT Uruguay

Facultad de Ingeniería

Hide & Seek

Robot de búsqueda BLE

Ignacio Lanzani - 143289

Luis Ignacio Porras - 186060

Pierina Santi- 201380

Tutores:

MSC. André Fonseca

Ing. Ilán Cohn

2020

ÍNDICE

1. Introducción	3	4.1. Diagrama de bloques	12
2. BLE	3	5. Ensamblaje y construcción del robot	13
3. Hardware	4	6. SOFTWARE	13
3.1.1. FC-51	4	6.1. Diagrama de flujo	13
Introducción	5	6.2. Explicación del programa	13
Principio de Funcionamiento	5	6.3. APP	14
Transmisor y receptor de infrarrojos	5	6.4. ALGORITMOS	14
Configuración de pines y diagrama		6.4.1. Desplazamiento	14
electrónico	5	6.4.2. Mapa	15
Dentro del FC -51	6	6.4.3. Decidir	15
Resistencia PULL UP	6	6.4.4. MedirRSSI	15
LM393	6	6.4.5. Cerebro, pseudo código	16
3.1.2. HC-SR04P	6	6.5. Dificultades	17
Cómo funcionan los sensores ultrasónicos	6	6.5.1. RSSI y múltiple conexión:	17
Características	7	6.5.2. Necesidad de precisión	17
Pines	7	6.5.3. Estructura	17
3.1.3. Conclusiones de los Sensores	7	6.5.4. Desarrollo	18
3.2. Pantalla	8	6.6. Conclusiones:	18
3.2.1 Introducción	8	6.7. Trabajo a Futuro:	18
OLED vs LED	8	7. Bitacora	18
PINES	8	9. Anexos	20
I2C vs SPI	8	9.1. Bibliografía	20
3.2.2 .Conclusión	9	9.2. Gestión del proyecto (Diagrama de Gantt)	21
3.3. Motores y Drive	9	9.3. Estructura del Robot	22
¿Qué es PWM?	9	Etapa 0:	23
3.3.1. Driver L298N	9	Etapa 1:	23
L298N	10	Etapa 2:	23
Conexiones y Funcionamiento	10	Etapa 3 (final):	23
3.3.2. Motor Reductor	10	Estructura final	24
3.3.3. Conclusiones:	11	9.4. CÓDIGO	25
3.4. ESP32	11	CALLBACKS	26
3.4.1. Introducción	11	EVENTOS	30
3.4.2. Características principales:	12	Setup	31
3.4.3. Esquema de pines:	12	Main	32
3.4.4. Conclusión	12	9.4.1. Librería	34
3.5. Bateria	12	SENSORES	35
4. CONEXIONES	12	FUNCIONES EXTRAS	37
		FUNCIONES Display	38
		FUNCIONES IMAGEN	39

FUNCIONES Deep Sleep	42
FUNCIONES MOTOR	42
FUNCIONES MOVIMIENTO	43
FUNCIONES DE BÚSQUEDA	45
CEREBRO	48
9.4.2. HEADERS	49
9.4.3. APP Mit Inventor	90
9.5. POSTER	95
9.6. Comandos	96
9.7. Diagrama de flujo	98
9.8 Link Al video	99
9.9 Link A Poster	99

Abstract- Bluetooth Low Level Energy es una PAN (Personal Area Network) inalámbrica su introducción en la industria apunta principalmente a los productos de fitness, salud y en los últimos años beacons. Al día de hoy la mayoría de los dispositivos móviles incluyen esta tecnología, que a diferencia de su iteración anterior (ahora conocido como Bluetooth Classic), mantiene un rango de señal similar, menor consumo energético , define nuevos protocolos de comunicación y funcionalidades que no eran posibles con anterioridad. Una de estas es la capacidad de medir la intensidad de la señal recibida (RSSI). Esta ultima es la que nos llevo a elegir el microcontrolador ESP32 (sucesor del ESP8266) como herramienta de desarrollo para nuestro robot, ya que entre sus múltiples prestaciones utiliza BLE.

1. Introducción

Arduino, en los últimos tiempos se ha convertido en no solo una empresa de hardware si no que su software cada vez incluye más periféricos y microcontroladores. Así es que desde hace un tiempo, el ESP32, al igual que su predecesor, son en su mayoría implementados a través de Arduino IDE. Programar en este microprocesador no es particularmente fácil, y es considerado por la mayoría de la comunidad (GitHub y espressif en particular) como una tarea “dura”. Arduino provee una manera rápida y eficaz de implementar muchas funcionalidades desarrolladas por la comunidad, acelerando el tiempo de desarrollo. Aun así, durante el transcurso de este proyecto nos fue imposible escapar de un entendimiento más profundo de sus librerías oficiales.

En este proyecto diseñamos y construimos un robot que utiliza BLE como método de comunicación con el usuario a través de una APP creada con MIT inventor. Este es capaz de desplazarse y “encontrar” al usuario que utiliza dicha app, de manera autónoma , gracias a que podemos medir la señal desde el celular. El proyecto fue desarrollado en ambas plataformas Arduino IDE y MIT app Inventor a partir de un microcontrolador ESP32. Como parte de la consigna el robot debía ser construido con materiales reciclables.

2. BLE

El Bluetooth de baja energía (Bluetooth Low Energy o BLE), es un subconjunto del estándar Bluetooth v4.0. Dispone de una pila de protocolos en referencia a la capa OSI completamente nueva y orientada a conexiones sencillas en aplicaciones de muy baja potencia (dispositivos dependientes de batería o pila).

La pila de protocolos para Bluetooth Low Energy sigue la estructura definida en la Figura 1.

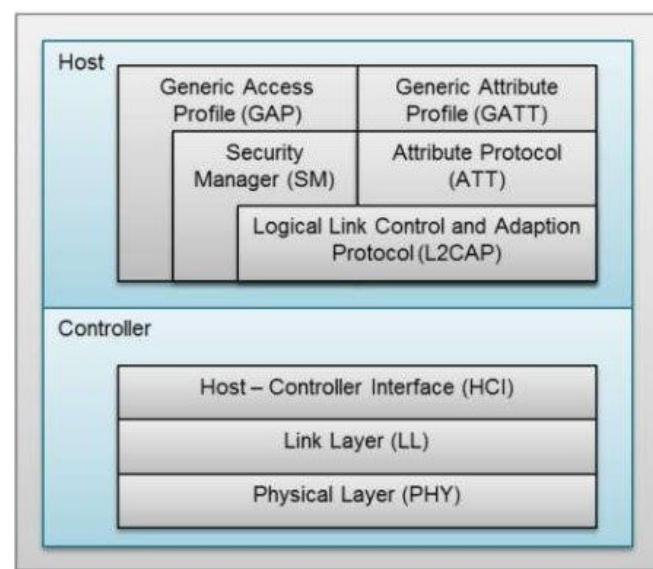


Figura 1 - Esquema Protocolo BLE(Tomado de smart-lightning.es [1])

La **capa física** es la encargada de realizar los procesos de modulación y demodulación de señales analógicas y posteriormente transformarlas en símbolos digitales. La tecnología BLE es capaz de utilizar hasta 40 canales de 2MHz en la banda ISM de 2.4 GHz (los canales 37, 38, 39 solo se usan para advertising, mientras que el resto es para transmisión de datos). El estándar emplea la técnica “frequency hopping” o “saltos en frecuencia”, siguiendo una secuencia de saltos pseudo-aleatorios entre los canales mencionados que ofrece un alto grado de robustez frente a interferencias.

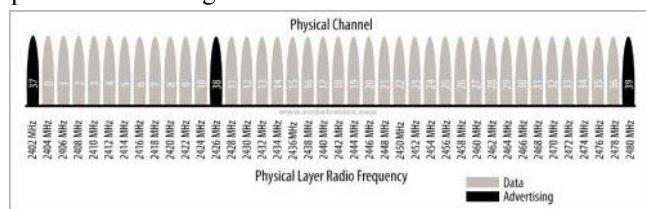


Figura 2 - Canales Físicos

La **capa de enlace** (link layer), se encarga de gestionar características como los requerimientos temporales del estándar, chequeo de mensajes y reenvío de mensajes erróneos recibidos, gestión, filtrado de direcciones etc. Además ofrece la definición de roles (Advertiser, Scanner, Master and Slave) que permiten identificar de forma lógica el rol de cada dispositivo en el proceso de comunicación. El nivel LL es del mismo modo responsable de procesos de control como el cambio de parámetros de la conexión o la encriptación.

HCI es un protocolo estándar que permite que la comunicación entre un host y un controlador se lleve a cabo a través de un interfaz serie. A modo de ejemplo, en la mayoría de smartphones u ordenadores el host y la aplicación corren en la CPU principal mientras que el controlador está situado en hardware específico y separado, conectado mediante UART o USB. El estándar Bluetooth define HCI como el conjunto de comandos y eventos para la interacción de ambas partes (host y controlador).

La **capa L2CAP** (Logic Link Control and Adaptation Protocol), se responsabiliza de dos tareas fundamentales en un proceso de comunicación. En primer lugar, el proceso de multiplexación, es decir, la capacidad de dar formato a mensajes provenientes de las capas OSI superiores y encapsularlos en paquetes estándar BLE así como el proceso inverso.

Para BLE, la capa **L2CAP** es la encargada de dar acceso y soporte a los dos protocolos fundamentales. Por un lado, **ATT** (Attribute Protocol), un protocolo basado en atributos presentados por dispositivo, con arquitectura cliente-servidor, que permite el intercambio de información. Por otro lado, **SMP** (Security Manager Protocol), protocolo que proporciona un framework para generar y distribuir claves de seguridad entre dos dispositivos.

En el nivel más alto de la capa de protocolos, encontraremos de forma paralela las capas GAP y GATT. Esta primera, **GAP** (Generic Access Profile), permite que un dispositivo sea visible para el resto de dispositivos y además determina cómo puede interactuar un dispositivo entre otro. Establece distintas normas y conceptos para estandarizar las operaciones de más bajo nivel como:

- Roles de interacción
- Modos de operación y transición entre ellos
- Procedimientos para establecimiento de comunicación
- Modos de seguridad y procedimientos

GATT (Generic Attribute Profile) por otra parte, define como dos dispositivos BLE transfieren información. Este proceso tiene lugar cuando dos dispositivos han superado la fase de establecimiento de comunicación (controlada por

GAP) y comienza la transferencia de información pudiendo ser de forma bidireccional.[1]

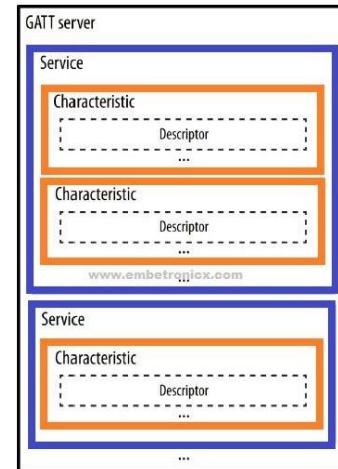


Figura 3 - GATT Server(Tomada de smart-lighthing.es[1])

RSSI es un indicador de intensidad de señal recibida (RSSI), en otras palabras es una medida estimada de lo bien que un dispositivo puede “oír”, detectar y recibir señales es equivalente al utilizado para redes WIFI. Este valor de señal se mide en decibelios desde 0 (cero) hasta -120. Cuanto más se acerque el valor a 0 (cero), más fuerte será la señal. Un RSSI de -55 es una señal más fuerte que -70.

3. Hardware

En esta sección describiremos el comportamiento y funcionamiento de los sensores utilizados en este proyecto, asimismo nuestras conclusiones sobre los mismos dada nuestra experiencia durante este proceso.

3.1. Sensores

3.1.1. FC-51

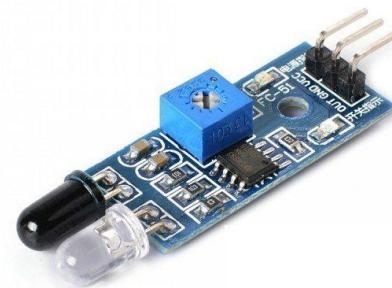


Figura 4 - FC-51[2]

Introducción

Mediante un sistema de rayos infrarrojos compuesto por un transmisor y un receptor, este sensor permite detectar la presencia de objetos frente a él. El mismo funciona enviando una señal de longitud de onda particular con un transmisor la cual es compatible con el receptor del sensor.

Existen distintos modelos de sensores infrarrojos para distintos tipos de aplicaciones. La tecnología IR se utiliza, por ejemplo, en sensores de proximidad para detectar un objeto en las inmediaciones, en sensores de contraste para identificar una trayectoria trazada en el suelo, o en sensores de conteo para contar los objetos que pasan frente al sensor.

Principio de Funcionamiento

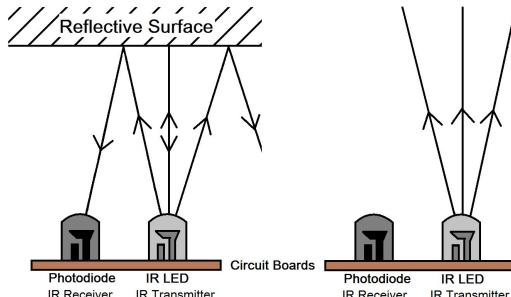


Figura 5 - Principio de funcionamiento[2]

El LED IR envía una señal que al llegar a una superficie reflectante, "rebota" en varias direcciones para luego ser captada por un fotodiodo y esto es indicado en uno de los pines. Cuando la superficie es de color blanco, el sensor la detecta mejor.

En el caso de una superficie absorbente (por ejemplo, de color negro), la señal se refleja menos y por tanto, el sensor apenas la detecta.

Transmisor y receptor de infrarrojos

El transmisor del sensor es un LED IR, este funciona igual a un LED común pero trabaja a la longitud de onda correspondiente a los rayos infrarrojos. El voltaje de trabajo es de 3.3-5 V y el consumo de corriente es de aproximadamente 20 mA. El receptor (un fotodiodo) es capaz de detectar la radiación infrarroja emitida por el transmisor. Estéticamente es similar a un LED pero la cápsula exterior se puede envolver en una película de color oscuro.

Configuración de pines y diagrama electrónico

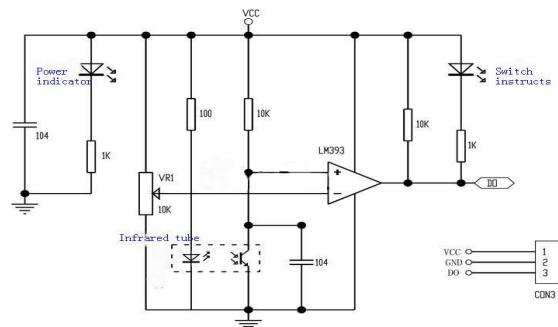


Figura 6 - Diagrama de un sensor IR FC-51[2].

El circuito integrado LM393 es un comparador de voltaje de colector abierto que proporciona una salida si hay una resistencia pull-up entre la salida del IC (DO) y la fuente de alimentación Vcc ($R = 10\text{ k}\Omega$): la salida de OD es alta si el objeto no es detectable, baja si el objeto es detectable.

El componente tiene tres pines de conexión:

1. Vcc para fuente de alimentación de 3.3-5V DC;
2. Gnd para la referencia a la masa;
3. Out para la señal de salida del sensor digital.

El sensor cuenta con dos LEDs, uno indica si el dispositivo se encuentra encendido y el otro indica cuando se detecta la presencia de un obstáculo.

La salida del sensor es alta (ALTA) si no se detecta un obstáculo (el receptor LED no recibe señales reflejadas), es baja (BAJA) en caso de un obstáculo.

Este sensor detecta objetos a una distancia de entre 2 y 30 cm. Con el potenciómetro, es posible calibrar la sensibilidad del sensor para acomodarse según la aplicación y las condiciones ambientales (por ejemplo, brillo). Al girar el potenciómetro en sentido antihorario, la distancia a la que el sensor detecta el objeto disminuye, al girarlo en sentido horario esta distancia aumenta.[2]

Dentro del FC -51

Resistencia PULL UP

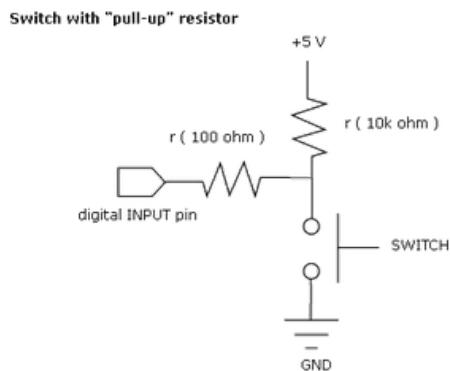
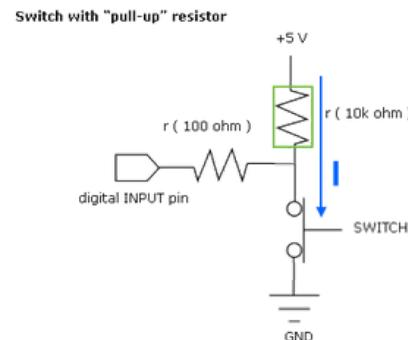
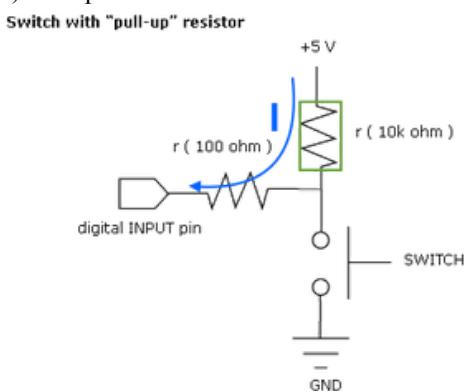


Figura 7 - Resistencia pull up(Tomada de [learningaboutelectronics.com\[3\]](http://learningaboutelectronics.com[3]))

La acción de Pull-Up en electrónica se asigna a la acción de elevar una tensión de entrada o salida que tiene un circuito lógico mientras éste está en reposo. Esto evita que se hagan lecturas erróneas si este pin ya no se encuentra conectado o no está recibiendo una señal. La resistencia se conecta a la fuente de alimentación. Cuando el interruptor está abierto la corriente va desde la fuente de alimentación al Vout dando un valor lógico de HIGH (Dibujo superior) y cuando el interruptor está cerrado la corriente se mueve hacia tierra (GND) dejando un 0 (LOW) en el pin.



Figuras 8 y 9 - Diagrama funcionamiento pull-up (Tomada de [learningaboutelectronics.com\[3\]](http://learningaboutelectronics.com[3]))

LM393

El LM393 es un comparador diferencial dual, esto significa que acepta 2 entradas para comparar. Compara estas entradas de voltaje y determina cuál es el valor mayor. En base a esto, se pueden tomar decisiones electrónicas.[4]

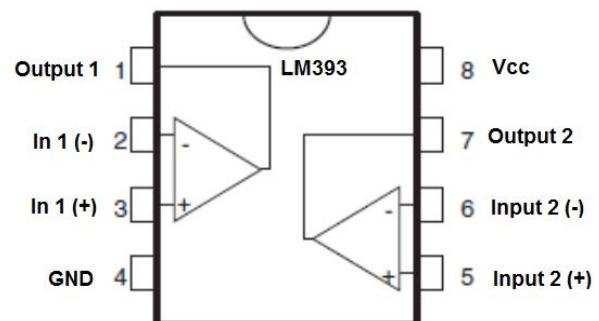


Figura 10 - Pinout LM393(Tomada de [learningaboutelectronics.com\[4\]](http://learningaboutelectronics.com[4]))

3.1.2. HC-SR04P

Cómo funcionan los sensores ultrasónicos

Los sensores ultrasónicos utilizan sonido a alta frecuencia para determinar la distancia que los separa del objeto más cercano frente a ellos.

El sensor envía una onda de sonido a una frecuencia específica y cuanta el tiempo que pasa hasta que la misma retorna (luego de haber rebotado en un objeto).

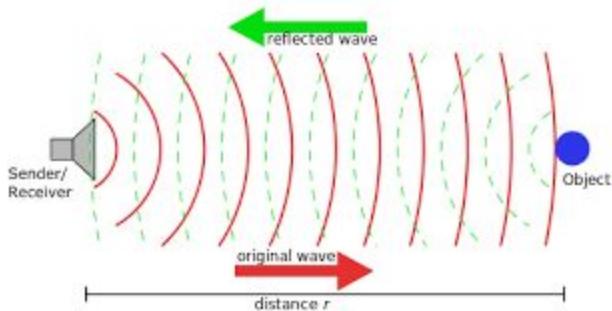


Figura 11 - Funcionamiento (Tomada de Elijah J. Morgan[5])

Cabe señalar que los sensores ultrasónicos tienen un cono de detección, el ángulo de este cono varía con la distancia, la siguiente figura muestra esta relación. La capacidad de un sensor para detectar un objeto también depende de la orientación de los objetos al sensor. Si un objeto no presenta una superficie plana al sensor, entonces es posible que la onda de sonido rebote en el objeto de manera que no regrese al sensor.



Figura 12 - Forma incorrecta de manipulación (Tomada de Elijah J. Morgan[5])

Características

- Fuente de alimentación: + 5V DC
- Corriente de reposo: <2 mA
- Corriente de trabajo: 2.8 mA en 5V
- Ángulo efectivo: <15°
- Distancia de alcance: 2 - 400 cm
- Ángulo de medición: 30°
- Ancho de pulso de entrada del disparador: 10uS
- Dimensión: 45 mm x 20 mm x 15 mm
- Peso: aprox. 8.5 g

Pines

El HC SR04P tiene cuatro pines, VCC, GND, TRIG y ECHO; todos estos pines tienen diferentes funciones. Los pines

VCC y GND son los más simples, que encienda el HC SR04. Estos pines deben conectarse a una fuente de 5 voltios y a tierra, respectivamente. Hay un pin de control único: el pin TRIG, este es responsable de enviar el pulso de ultrasonido. El trigger debe establecerse en ALTO durante 10 μ s, en este momento el HC SR04P emite una ráfaga sónica de ocho ciclos a 40 kHz. Después de que se haya enviado esta ráfaga, el pin ECHO irá ALTO. Este es responsable de detectar el rebote de la ráfaga y por ende tomar las distancias. EL pin se mantendrá en ALTO hasta que sea momento de enviar una nueva ráfaga. [5]

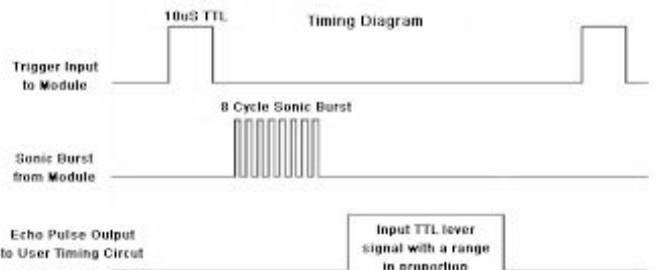


Figura 13 - Pulsos del Sensor (Tomada de Elijah J. Morgan[5])

3.1.3. Conclusiones de los Sensores

La conclusión principal a destacar, es que a pesar de que el FC 51 es más conveniente a nivel económico, en cuanto a funcionamiento el HC-SR04P es superior.

A pesar de que el sensor IR tiene la ventaja de poder calibrarse, notamos que cada pequeño movimiento en el potenciómetro empieza a dar errores de medidas, por lo que decidimos tocarlo lo menos posible, lo cual limita mucho la experiencia de uso. Otra desventaja es que, al ser un sensor de luz hay obstáculos que no logra detectar de manera correcta. Esto implica que si colocamos estos sensores en la parte frontal del robot, no detectaría todos los obstáculos del camino y en consecuencia, el robot chocaría contra ellos. También cabe detectar que el consumo de estos sensores es bastante elevado (30mA), lo que hace que no podamos tener esos sensores prendidos al mismo tiempo.

En cuanto al sensor ultrasónico, la mayor dificultad que se nos presentó fue a la hora de agregarlo a la estructura del robot, ya que son considerablemente más grandes que los IR. Notamos que este sensor funciona bien al frente del robot, ya que es capaz de sensar más obstáculos que el IR y de forma más precisa. Además, como este sensor es de bajo consumo (2mA), podemos mantenerlo encendido todo el tiempo.

Cabe destacar, que notamos que ambos sensores tienen ángulos de trabajo muy pequeños y por ende el ángulo con el

que el robot se aproxima hacia el objeto es clave para poder ser detectado.

Tomando en cuenta todo lo destacado anteriormente, decidimos colocar dos sensores ultrasónicos en el frente para "expandir" el ángulo de detección; además decidimos colocar un sensor IR en cada lateral del robot para proporcionar información extra a la hora de realizar giros.

3.2. Pantalla

3.2.1 Introducción

OLED vs LED

Las pantallas LED LCD utilizan una luz de fondo para iluminar los píxeles mientras que los píxeles OLED producen su propia luz, lo que significa que el brillo de una pantalla OLED puede ser controlado pixel por pixel.

4.2.1. OLED 7 pines 0.96 pulgadas

Al igual que cualquier pantalla necesita un controlador específico que convierta los datos en señales electrónicas, nuestro modelo utiliza el SSD1306. Para programarla en Arduino son necesarias las siguientes librerías:

- Adafruit_SSD1306
 - Adafruit_GFX
- Características:
- Voltaje de alimentación 3,3V - 5V
 - Consumo de corriente: 20mA
 - Resolución: 128*64 pixels
 - Controlador: SSD1306
 - Tipo de comunicación: I2C-SPI (Nuestro caso SPI)

Decidimos utilizar una pantalla de este estilo por su facilidad y para poder visualizar los distintos estados y datos de relevancia durante la pruebas.

PINES

1. Ground(Gnd): Se conecta a la tierra del circuito.
2. Supply(Vcc): funciona desde 3.3 a 5V.
3. SCK(D0): La pantalla es compatible con I2C y SPI, cuyo reloj se suministra a través de este pin.
4. SDA(D1): Este es el pin de datos, se puede usar para I2C o para SPI.
5. RESET(RST): Cuando se mantiene a tierra momentáneamente, este pin restablece el módulo.
6. DC: Es el pin de comando, se puede usar para I2C o SPI.
7. Chip Select(CS): Normalmente se mantiene bajo, se usa solo cuando más de un dispositivo SPI está conectado al MCU (microcontrolador).[6]

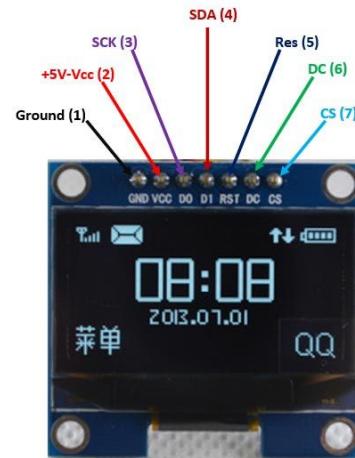


Figura 14 - Pinout OLED SSD1306

I2C vs SPI

Ventajas I2C:

- Eficiente, porque permite la comunicación entre múltiples maestros y múltiples esclavos.
- Permite la comunicación utilizando solo dos cables.
- Utiliza el bit ACK / NACK que confirma que cada trama de datos se ha transmitido correctamente.

Desventajas I2C:

- I2C tiene una interfaz semidúplex.
- Puede volverse complejo a medida que aumenta el número de dispositivos.
- Puede volverse complejo con un número creciente de dispositivos.

Ventajas SPI:

- No hay bit de inicio y parada, por lo que los datos se pueden transferir continuamente sin interrupciones.
- SPI es un protocolo de comunicación full-duplex.
- Dispone de bus de datos de alta velocidad de 10 MHz.
- Diferentes líneas MISO y MOSI, para que los datos se puedan enviar y recibir simultáneamente.

Desventajas SPI:

- SPI utiliza cuatro cables para la comunicación.
- Solo admite un maestro.
- Se necesita un pin maestro dedicado adicional para CS o SS para cada esclavo adicional.
- No se establece ningún protocolo para la detección de errores.

- No se establece ningún mecanismo de acuse de recibo y, por lo tanto, no se confirma la recepción de datos.[7]

3.2.2 .Conclusión

Teniendo en cuenta las ventajas y desventajas presentadas anteriormente, decidimos utilizar el SPI. Esta es la opción que nos trae menos complicaciones, ya que es la que venía por defecto y contamos con la cantidad de pines necesarios para hacerla funcionar. Como extra, viene con mayor velocidad de transmisión la cual nos permite crear mejores animaciones.

3.3. Motores y Drive

Para nuestro modelo, nos basamos en una estructura que contiene dos motores reductores que son controlados mediante un driver, utilizando PWM.

¿Qué es PWM?

Modulación por Ancho de Pulso o PWM es un tipo de señal de voltaje utilizada para enviar información o modificar la cantidad de energía enviada a una carga. En general, se utiliza cuando en un circuito digital se necesita emular una señal analógica. El resultado es llamado duty cycle y sus unidades están representadas en términos de porcentaje. [8]

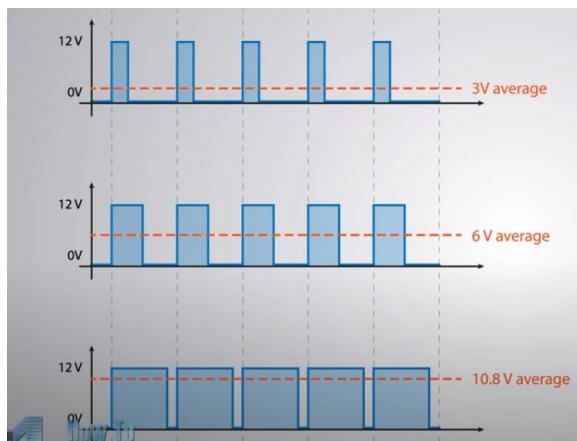


Figura 15 - PWM([Tomada de Google](#))

3.3.1. Driver L298N

¿Qué es un puente H? Un Puente H es un circuito electrónico generalmente usado para permitir a un motor eléctrico DC girar en ambos sentidos (avance y retroceso). Son ampliamente usados en robótica y están disponibles en su mayoría como circuitos integrados.

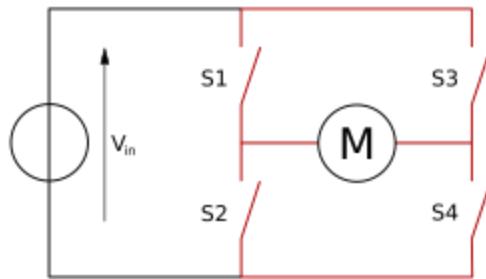


Figura 16 - Ilustración forma puente H([Tomada de wikipedia](#))

El término "puente H" proviene de la representación gráfica del circuito. Un puente H se construye con 4 interruptores (mecánicos o mediante transistores). Cuando los interruptores S1 y S4 están cerrados (y S2 y S3 abiertos) se aplica una tensión positiva en el motor, haciéndolo girar en un sentido. Abriendo los interruptores S1 y S4 (y cerrando S2 y S3), el voltaje se invierte, permitiendo el giro en sentido inverso del motor.

Con la nomenclatura usada, los interruptores S1 y S2 nunca podrán estar cerrados al mismo tiempo, porque esto cortocircuitaría la fuente de tensión. Lo mismo sucede con S3 y S4.

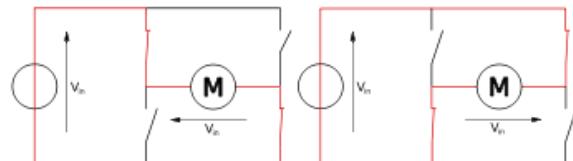


Figura 17 - Posibles Movimientos ([Tomada de wikipedia](#))

A continuación incluimos una figura sobre cómo configurar los motores.[9]

S1	S2	S3	S4	Resultado
1	0	0	1	El motor gira en avance
0	1	1	0	El motor gira en retroceso
0	0	0	0	El motor se detiene bajo su inercia
1	0	1	0	El motor gira derecha
0	1	0	1	El motor gira izquierda
1	1	0	0	Cortocircuito
0	0	1	1	Cortocircuito
1	1	1	1	Cortocircuito

Figura 18 - Tabla de Movimientos(Tomada de Wikipedia)

L298N

El módulo controlador de motores L298N H-bridge nos permite controlar la velocidad y la dirección de dos motores de corriente continua, gracias a los 2 dos H-bridge que monta.

El rango de tensiones en el que trabaja este módulo va desde 3V hasta 35V, y una intensidad de hasta 2A. Cuando lo alimentamos hay que tener en cuenta que la electrónica del módulo consume unos 3V, así que los motores reciben 3V menos que la tensión con la que alimentamos el módulo.

Además el L298N incluye un regulador de tensión que nos permite obtener del módulo una tensión de 5V, la cual nos sirve para conectar la ESP 32. Hay que tener en cuenta que este regulador sólo funciona si se alimenta el módulo con una tensión máxima de 12V.

Conecciones y Funcionamiento

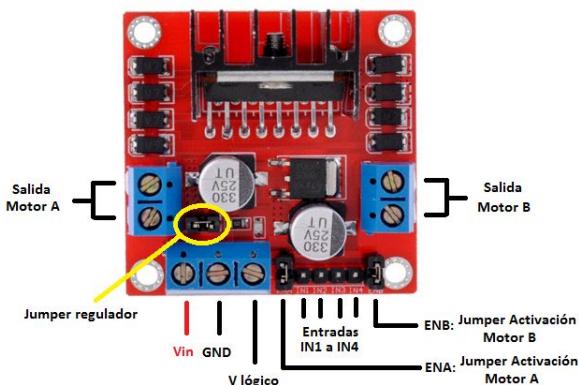


Figura 19 - Pinout L298N

La entrada de tensión Vin admite tensiones entre 3V y 35V, y justo a su derecha en la imagen tenemos el pin que debemos conectar a GND.

La tercera conexión de ese grupo V lógico puede funcionar de dos maneras:

1. Si el jumper del regulador está cerrado activaremos el regulador de tensión de la placa que contiene al L298N, y en V lógico tendremos una salida de 5V, que podremos usar para lo que queramos, por ejemplo para alimentar una placa Arduino.
2. Si le quitamos el jumper se desactiva el regulador, necesitando alimentar la parte lógica del módulo de manera externa. En este caso, tendremos que colocar una tensión de 5V por la conexión V lógico para que el módulo funcione.

Observaciones: Si introdujeremos corriente por V lógico con el jumper de regulación puesto podríamos dañar el

módulo. Además el regulador sólo funciona con tensiones hasta 12V en Vin, por encima de este valor tendremos que quitar el jumper y alimentar la parte lógica del módulo desde otra fuente.

Las salidas para los motores A y B les (IN_i, i=1,2,3,4) brindarán la alimentación necesaria para moverse. Hay que tener en cuenta la polaridad al conectarlos, para que cuando se mueven lo hagan en el sentido esperado. En caso de que esto no suceda, bastará con invertir la conexión. Los pines IN1 e IN2 controlan el sentido de giro del motor A y los pines IN3 e IN4 el del motor B. Funcionan de forma que si IN1 está a HIGH e IN2 a LOW, el motor A gira en un sentido y si está IN1 a LOW e IN2 a HIGH lo hace en sentido contrario. Lo mismo con los pines IN3 e IN4 del motor B.

Para controlar la velocidad de giro de los motores tenemos que quitar los jumpers y usar los pines enable (ENA y ENB). Los conectaremos a dos salidas PWM del ESP 32 el cual enviará un pulso de valor entre 0 y 255 que controle la velocidad de giro. Si tenemos los jumpers colocados, los motores girarán siempre a la misma velocidad.[10]

En nuestro caso las conexiones serán algo similar a esto:

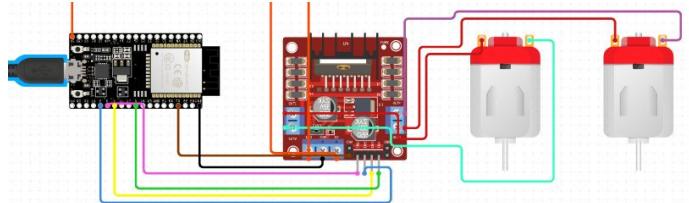


Figura 20 - Esquema de conexión(Imagen creada con circuit.io)

3.3.2. Motor Reductor

Analizamos el principio de funcionamiento de los motores. Supongamos que la rueda "A" de la fig.1 tiene un diámetro de 5 cm, su perímetro será entonces de $5 \times 3.1416 = 15.71$ cm. El perímetro es la longitud total del envolvente de la rueda. Una rueda "B" de 15 cm de diámetro y 47.13 cm de perímetro (15×3.1416) está haciendo contacto con el perímetro de la rueda "A" (fig 2)

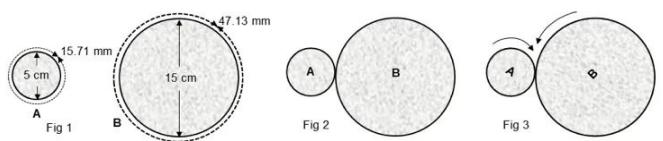


Figura 21 - Ejemplo funcionamiento Motorreductor

En la fig 21.3, cuando gira la rueda "A" hará que a su vez gire la rueda "B" pero sucederá que por cada tres vueltas que dé "A", la rueda "B" solamente dará una vuelta, esto es, el diámetro de "B" dividido por el diámetro de "A" ($15/5 = 3$).

Este número 3 será la relación de reducción de este reductor o motorreductor elemental y se indica como 3:1.

Con esta simple combinación se ha logrado disminuir la velocidad de rotación de la rueda “B” a la tercera parte de la velocidad de la rueda “A”. Si a la combinación de ruedas antes descrita encadenamos más ruedas adicionales iremos consiguiendo una velocidad menor cada vez hasta alcanzar la deseada (6:1, 30:1, 100:1). También se pueden alcanzar velocidades muy pequeñas de ser necesario de forma que, por ejemplo, la rueda “A” tuviera que girar cientos de veces para que la última rueda girara una sola vez. En este caso tendremos un motorreductor de varios trenes de reducción, entendiendo como 1 tren de reducción a un par de ruedas. Con 6 ruedas tendríamos tres trenes de engranes.

Con este sistema de reducción no solamente disminuimos la velocidad de “B” a un giro más lento sino que al mismo tiempo estaremos aumentando el “par” o “torque” en la última rueda del motorreductor. Esta rueda se conoce como la rueda de salida a la que va ensamblada la “flecha de salida” del reductor o motorreductor.[11]

En nuestro caso en el mercado existe el modelo 48:1 que tiene la siguiente ficha técnica:

Voltaje de Operación	DC 3V	DC 5V	DC 6V
Parámetros Caja Reductor	Reducción	48:1	
Velocidad sin carga	125 RPM	200 RPM	230 RPM
Velocidad con carga	95 RPM	152 RPM	175 RPM
Torque de salida	0.8kg.cm	1.0kg.cm	1.1kg.cm
Velocidad del robot sin carga (metros/minuto)	25.9	41.4	47.7
Corriente	110-130mA	120-140mA	130-150mA
Diámetro máximo de llanta	6.5cm		
Dimensiones	70mm x 22mm x 18mm		
Peso	50g		
Ruido	<65dB		



Figura 22 - Motorreductor (Tomada de patagoniatec.com[11])

3.3.3. Conclusiones:

Al inicio de este proyecto, se investigaron diversos tipos de motores para ver cuál se ajustaba más a nuestras necesidades.

La decisión final fue tomada teniendo en cuenta el factor económico así como el peso de la estructura final y el torque del motor. Además, decidimos que sería más fácil utilizar dos motores y un driver para darle movimiento a nuestro proyecto.

El centro de masa del robot se encuentra en la parte delantera del mismo, siendo la batería de alrededor de 700g. Los motores elegidos, tienen un torque de 1,1 Kg por cm, y nuestras ruedas son de aprox 3 cm. Así es que determinamos que estos funcionaran ya que necesitamos 1kg/cm para mover el robot correctamente, si no consideramos una distribución más simétrica sobre la superficie de apoyo de las partes del robot sobre las ruedas. Este estimativo nos da un margen por cualquier eventualidad.

3.4. ESP32

3.4.1. Introducción

Creado por Espressif Systems, el ESP32 es un sistema de bajo consumo y bajo costo en un chips SoC (System On Chip) con Wi-Fi y modo dual con Bluetooth (BLE). En el fondo, hay un microprocesador Tensilica Xtensa LX6 de doble núcleo o de solo un núcleo con una frecuencia de reloj de hasta 240 MHz.

El ESP32 está altamente integrado con switch de antena , balun para RF, amplificador de potencia, amplificador de recepción con bajo nivel de ruido, filtros y módulos de administración de energía, totalmente integrados dentro del mismo chip.

Diseñado para dispositivos móviles; tanto en las aplicaciones de electrónica, y las de IoT (Internet de las cosas), ESP32 logra un consumo de energía ultra bajo a través de funciones de ahorro de energía incluye la sintonización de reloj con una resolución fina, modos de potencia múltiple y escalado de potencia dinámica.[12]

3.4.2. Características principales:

- Procesador principal: Tensilica Xtensa LX6 de 32 bits.
- Wi-Fi: 802.11 b / g / n / e / i (802.11n @ 2.4 GHz hasta 150 Mbit / s).
- Bluetooth: v4.2 BR / EDR y Bluetooth Low Energy (BLE).

- Frecuencia de Clock: Programable, hasta 240MHz.
 - Rendimiento: hasta 600DMIPS(1).
 - ROM: 448KB, para arranque y funciones básicas.
 - SRAM: 520KB, para datos e instrucciones.

1. **DMIPS:** DMIPS es el acrónimo de Dhrystone Millions of Instructions Per Second, o sea, Millones de Instrucciones Por Segundo de tipo Dhrystone(2).[13]
 2. El **Dhrystone** es un pequeño benchmark(prueba de rendimiento) sintético que pretende ser representativo de programación entera de sistemas. Está basado en estadísticas publicadas sobre uso de particularidades de los lenguajes de programación, sistemas operativos, compiladores, editores, etc.[14]

3.4.3. Esquema de pines:

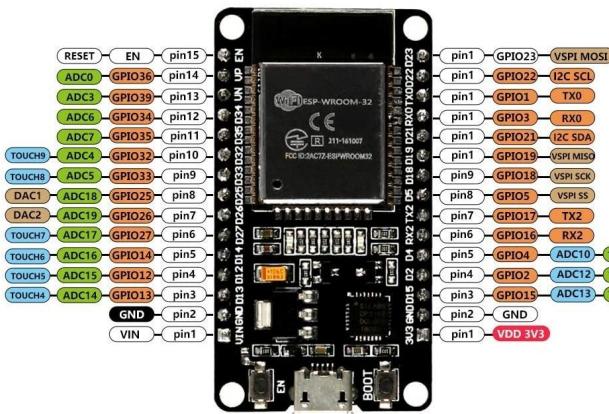


Figura 23 - Esquema de pinos (Tomada de Google)

3.4.4. Conclusión

Decidimos utilizar esta placa debido a que nuestro proyecto se basa en la tecnología BLE para encontrar a un usuario a través del valor de RSSI y el ESP 32 nos permite implementarlo.

Además, esta placa es compatible con los sensores que decidimos utilizar y es más rápida que un Arduino UNO. La placa UNO maneja una frecuencia de 16MHz vs los 240MHz del ESP 32.

3.5. Bateria

Consideremos el consumo de cada uno de los componentes del robot:

- sensor de proximidad =30mA),
 - HC-SR04p =(2mA)x2,
 - motores = (150mA) x2,

- panta =20mA
 - esp32=50mA

Así, teniendo en cuenta las opciones en plaza, y las capacidades de nuestros componentes (principalmente el driver), determinamos que la batería de 12V y de 1,2 Ah nos iba a permitir utilizar al robot por un tiempo prolongado brindando la cantidad de corriente necesaria para todo el sistema, está dentro del rango de voltaje del driver y brindará a los motores del voltaje máximo que necesitan.

4. CONEXIONES

4.1. Diagrama de bloques

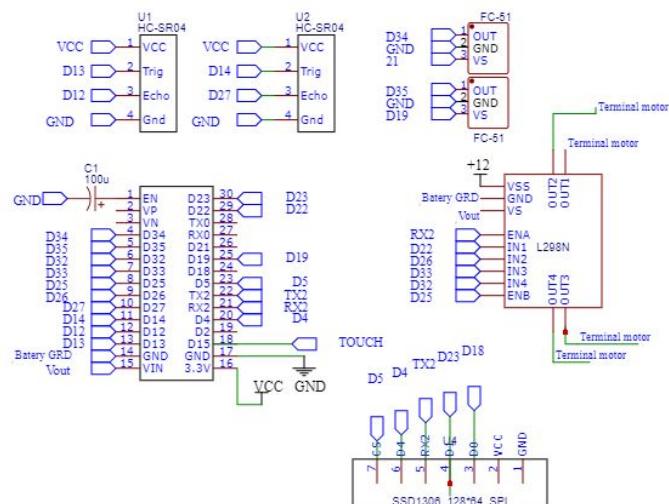


Figura 24 - Diagrama de Bloques

Nota: en la *Figura 24* podemos ver un capacitor de 100u. Este nos permite subir el código al ESP32 de manera más rápida , consistente y sin errores.

5. Ensamblaje y construcción del robot

Para el diseño de nuestro robot nos basamos en el personaje principal de la película Wall-e. Teniendo en cuenta que la consigna del proyecto fomentaba el uso de materiales reciclables en la mayoría de la estructura del robot, decidimos utilizar cartón y plástico para su construcción. Esto puede observarse en el anexo 9.3.

Como base del robot, utilizamos cartón de una cuadernola de tapa dura, ya que este probó ser lo suficientemente firme para soportar el peso de los componentes así como el resto de la estructura. En esta base, tenemos los motores y ruedas de soporte, elementos que requirieron de una

precisa colocación para asegurar el correcto movimiento del robot.

Para las caras superiores del cubo, utilizamos cartón de distintas cajas, teniendo el cuidado de hacer las debidas perforaciones para cada componente. Las caras laterales, llevan los sensores infrarrojos y el botón de encendido, en la cara frontal, se encuentran los dos sensores de ultrasonido y en la cara posterior incluimos una “ventana” de papel celofán para ver la electrónica interna. Por último en la tapa superior, tenemos la pantalla oled y el ESP 32.

Originalmente, la pantalla iba a ir en la cara frontal, pero era muy difícil ver las animaciones proyectadas. Además, el ESP 32 iba a ir dentro de la estructura pero al dejarlo por encima de la misma mejora la lectura de valores RSSI. A su vez la construcción de la parte superior permite remover parte de la estructura dejando al microcontrolador en el lugar, facilitando las tareas de mantenimiento como puede observarse en la figura estructura superior expuesta del anexo 9.3.

Por último, decidimos agregar los característicos ojos y brazos de Wall-e para mejorar su estética. Los mismos fueron realizados con cartón.

Cabe destacar que debido a la naturaleza de los materiales elegidos tuvimos varias dificultades a la hora de diseño y ensamblaje del modelo. Nuestros problemas principales fueron a la hora de pegar la estructura y asegurar su estabilidad y durabilidad.

6. SOFTWARE

6.1. Diagrama de flujo

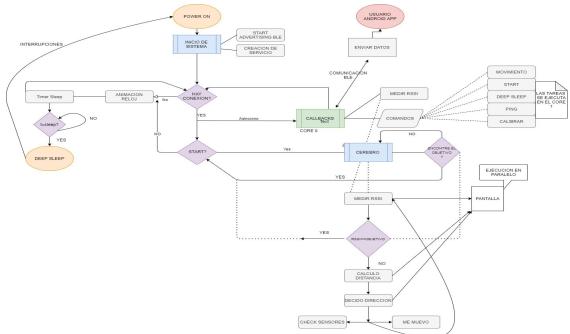


Figura 25 - Diagrama de flujo La versión ampliada se encuentra en el apéndice 9.5

6.2. Explicación del programa



Figura 26 - Animación de tiempo en inactividad.

El programa (anexo 9.4) está formado por distintas tareas en ejecución concurrente (Véase Anexo 9.4 Main), por lo cual utilizamos freertos, para poder utilizar los 2 cores del ESP32 de manera eficiente. Mientras se espera por la conexión del usuario, la animación de la *Figura 26*. Se presenta al usuario, si no hay acciones durante 4 minutos el robot entra en deepsleep (modo de ahorro energético). Puede ser despertado tocando la chapa touch (físicamente al lado del ESP32).

La comunicación entre el usuario y la ESP32 se trabaja de manera asincrónica dentro del core 0 (anexo 9.4 callbacks), que queda encargado de recibir los datos desde la APP del usuario. Así, una vez establecida la conexión medimos la RSSI desde el lado del usuario y lo enviamos de nuevo al ESP32, esto ocurre 5 veces por segundo. Esta velocidad tiene un punto crítico, en el cual los buffers y el stack se saturan. Utilizando la función de ping creada por nosotros, determinamos que 200ms por envío funcionaba de manera estable.

Es importante tener en cuenta que cualquier tipo de delay/sleep, dentro de una tarea puede afectar enormemente la estabilidad y precisión del programa. Así, es que se evita poner estos mismo dentro del callback , ya que este puede verse saturado si se detiene mientras la comunicación sigue activa.

El usuario desde su app puede enviar comandos (véase anexo 9.6) que tienen distintos efectos en el robot, algunos de estos son básicos como moverse adelante, prender la luz del led, o ponerlo a dormir. Los comandos más importantes son el start y la calibración. La calibración cumple la función de determinar que es “encontrar” al usuario. A través de esto, se guarda en memoria la posición que debe buscarse, si esta es muy cercana el robot se acercara pero nunca encontrará al usuario.

Al iniciarse el sistema de búsqueda, el robot procederá a moverse, indicará sus estados a través de animaciones. Este proceso se ejecuta en el core 1. Explicaremos el algoritmo de búsqueda más adelante.

Si no se registran acciones durante dos minutos , el robot entrará en un modo de deep sleep.

Las animaciones que se muestran en la pantalla se ejecutan en paralelo y son controladas por semáforos, ya que de no ser así, se superpondrán ejecuciones sobre el recurso compartido que en este caso es la pantalla, lo cual generará visuales indeseadas (glitches).

6.3. APP

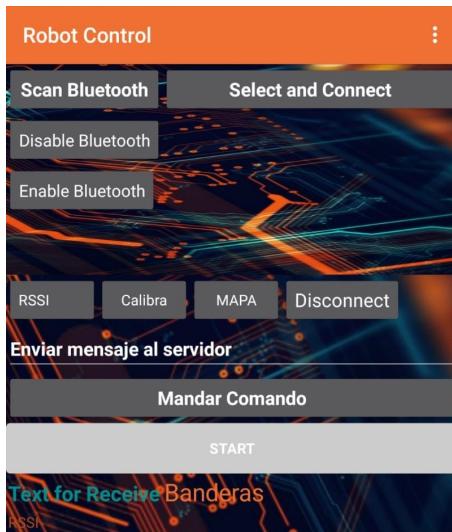


Figura 27 - Interfaz del usuario



Figura 28 - Interfaz del usuario

En la Figura 27 vemos la pantalla de inicio del usuario. Aquí el mismo, tiene que habilitar ubicación para utilizar esta app así como habilitar el bluetooth. Luego debe hacer un “scan” y en la lista de dispositivos aparecerá (de estar encendido y no durmiendo) el servidor “Hide&Seek”.

Con el envío de mensajes de texto , se pueden mandar los comandos que se encuentran en el apéndice. Tenemos acceso al mapa del robot, rssi medido, desconexión y calibración.

Las respuestas del server se ven en el campo Text For Recibir, en banderas aparecerá la mac del esp32, como vemos en la Figura 28 . Debido al tiempo de desarrollo es difícil afirmar que la APP funcionará en otros dispositivos móviles con la misma eficiencia, ya que MIT no permite escanear el hardware para ajustarlo según sus características.

Esta app fue desarrollada con MIT inventor.

6.4. ALGORITMOS

6.4.1. Desplazamiento



Figura 29 - Animación durante la búsqueda.

Creamos esta función (Figura 29) que relaciona el tiempo de desplazamiento del robot con el RSSI, creamos el algoritmo de desplazamiento (Véase Anexo 9.4.1 Funciones de movimiento), teniendo en cuenta nuestra experiencia durante el proyecto. Notamos que las mediciones en general nunca serán mayores a -30db y es raro encontrar mediciones cercanas a los -90db. Es así que dimos peso a los casos de RSSI más observados durante las pruebas. Parece lógica la elección de una ecuación cuadrática dado que estamos midiendo intensidad de señal. Aún así, para poder darle más peso a las zonas cercanas ($RSSI > -50$) agregamos un término de la forma $c/RSSI$, teniendo en cuenta que el signo de RSSI siempre es negativo, podemos disminuir el movimiento aún más cuando el robot está próximo al RSSI de calibración (desde ahora en más RSSIc).

$$a * RSSI + B * RSSI + \frac{C}{RSSI} = t$$

Figura 30 - Ecuación de tiempo en función de RSSI.

$$\begin{cases} a \cdot (-80)^2 - b \cdot (80) - \frac{d}{80} = 4.5 \\ a \cdot (-50)^2 - b \cdot (50) - \frac{d}{50} = 2.5 \\ a \cdot (-40)^2 - b \cdot (40) - \frac{d}{40} = 0.5 \end{cases}$$

Figura 31 - Determinando a,b y c.

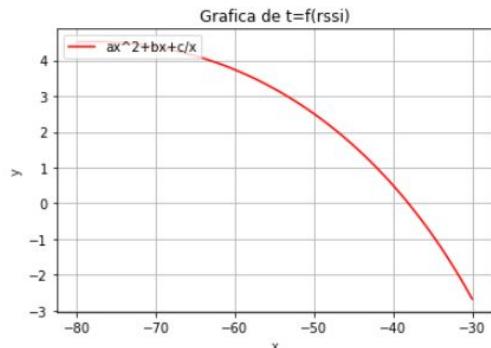


Figura 32 - Gráfica de $t=f(RSSI)$.

Así llegamos a que los valores de a,b y c planteados en la Figura 30 son:

- a=-0.00179
- b= -0.238
- c=246.

6.4.2. Mapa

El mapa , es un array de largo 4. En el que se marcarán las direcciones que se consideran buenas o malas según el resultado medido. Así, cuando se llame a la función de decisión esta mirará en el mapa todos los lugares que están marcados como negativos, para poder decidir qué dirección tomar. Una posición se marca únicamente cuando el RSSI medido es menor que RSSIc.

- Mapa[0]=adelante
- Mapa[1]=derecha
- Mapa[2]=atras
- Mapa[3]=izquierda

Véase Anexo 9.4.1 Funciones de búsqueda

6.4.3. Decidir

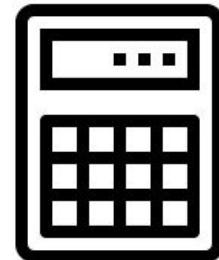


Figura 33 - Animación de espera antes de decidir.

El algoritmo de decisión, mirará el mapa y los sensores, esta función determinará en qué sentido se girará, y reseteará el mapa si este está lleno. Un ejemplo sería entrar al algoritmo con {1,1,0,0} , en este caso nuestra opción de giro será movernos a la izquierda. Aquí el mapa se rotará en ese sentido, con lo cual Mapa={0,1,1,0}. Entradas como {1,1,0,1} provocarán un giro de 180 grados. La animación de la Figura 33, se muestra antes de decidir para darle tiempo al robot de frenar correctamente antes del giro, logrando así mejores resultados (cercaos a los 90 grados).

Véase Anexo 9.2.1 Funciones de búsqueda

6.4.4. MedirRssi



Figura 34 - Animación de espera mientras se mide

La medición se da durante 6 segundos generando un total de 40 muestras. Si el resultado refleja una varianza alta entonces la medición se vuelve a ejecutar. Este algoritmo es recursivo y puede entrar en loop si las condiciones no son buenas. De todas maneras creemos es mejor reflejar este resultado , ya que el robot queda quieto en el mismo estado de medición indefinidamente, dejando en claro que no es posible su implementación. Recordar que el dispositivo que mide la señal es el celular desde la APP y reenvía esta en forma de dato al ESP32.Véase Anexo 9.4.1 Funciones de búsqueda.

6.4.5. Cerebro, pseudo código

Véase Anexo 9.4.1 Cerebro por loop actual y

Véase Anexo 9.4. Main para ver la llamada.

```

cerebro(mapa,rssi, sensores,obstacle){
  distancia = distanciaRssi(rssipromedio);
  while (rssi <RSSIc) {
    distancia = goForward(distancia, sensores,jugar);
    rssipromedio = medirRssi( 6seg);
    setRssiAnterior(rssipromedio);
    if((rssipromedio < rssianterior) ) {
      marcarMapa(mapa);
      moverAtras();
      decidir(mapa, sensores);
    }
    else if (obstacle) {
      moverAtras();
      decidir(mapa, sensores);
    }
    rssianterior = rssipromedio;
    distancia = distanciaRssi(rssipromedio);
  }
  println("te encontre y me canse");
}
  
```

En el código original, tenemos incluidas las animaciones y paradas necesarias para que los movimientos sean lo más precisos posibles.

Ejemplo:

El funcionamiento de búsqueda se basa en varios algoritmos que dependen exclusivamente del RSSI. Al iniciarse la búsqueda entraremos en la función “cerebro”, cuya primera acción es medir el RSSI en la posición “P1” para determinar así cuánto se desplaza hacia adelante. Supongamos que la RSSI de calibración (RSSIc) es “-42 db”. Como vemos en la *Figura 35*, el robot parte de (P1), y dependiendo de la medición (si es mayor, menor o igual al RSSIc) se desplazará hasta P2. En esta posición , se vuelve a medir la señal y se determina que su valor es “-60 db”. El robot entonces marcará en su mapa que esa no es la dirección correcta, se desplazará la misma cantidad en sentido contrario, que lo dejará en P1 de nuevo. Aquí, se chequearon los sensores y el mapa, para determinar cuál es la dirección de giro. En este caso nuestro mapa contiene {1,0,0,0} por lo tanto podemos girar izquierda , derecha o 180. El algoritmo de decisión siempre da prioridad al giro de la izquierda por diseño. Así en este caso giramos 90 grados en sentido antihorario y nuestro mapa ahora quedará {0,1,0,0}. Esto último indica que no debemos girar a la derecha en nuestro próximo movimiento, de tener el resto de las posibilidades libres (no obstáculo adelante o a la izquierda).

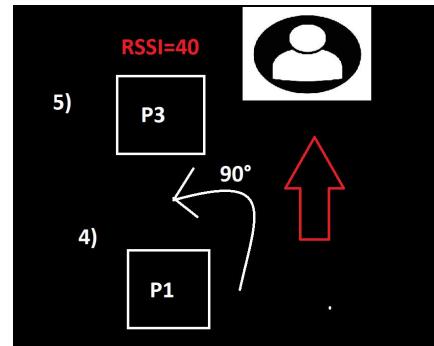


Figura 35 - Primera parte del movimiento.

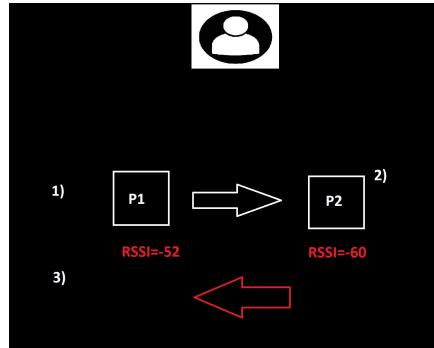


Figura 36 - Movimientos del robot.

Ahora *Figura 36*, la P1, determinara que el movimiento es ir de nuevo hacia adelante lo que llevará al robot a P3, aquí podran pasar dos cosas, el usuario puede acercar el celular al robot, así activando un evento de finalización, o el robot frena y medirá de nuevo (-40db). En ambos casos el mecanismo de búsqueda se dará por finalizado.

Si el RSSI medido hubiese sido mejor, pero no menor a RSSIc, el robot se moverá de nuevo hacia adelante, si pasa cerca del usuario y este está en una zona donde $\text{RSSI} \leq \text{RSSIc}$, el robot se detendrá antes de finalizar toda la trayectoria. Esto es gracias a que la medición de señal se hace de manera paralela y en otro procesador, con lo cual podemos acceder siempre a la última medida.

En caso de ser peor se marca el mapa y se llama al mecanismo de decisión.

6.5. Dificultades

6.5.1. RSSI y múltiple conexión:

Medir RSSI, hay varias maneras de obtener esta medida y a lo largo del proyecto se manejaron distintas funciones. Por un lado podemos obtener el RSSI , haciendo un SCAN de los dispositivos que estén en Advertising. La

desventaja de este proceso son varias, entre ellas que tenemos que escanear todos los dispositivos, y buscar el nuestro entre ellos. Cuando un dispositivo como un Android inicia la etapa de advertising utiliza una MAC aleatoria, esto es por un tema de seguridad y dificulta desde el lado del ESP32 identificar a los dispositivos.

Por otro lado las librerías para el ESP32 no tienen una función para medir el RSSI de los dispositivos conectados. Para poder llevar a cabo esta tarea tuvimos que emplear nuestra propia función, y registrar la dirección MAC en el momento de la conexión utilizando las funciones de espresif. Finalmente pedir el RSSI, llamando a un evento del protocolo gap, el cual lo devuelve entre varios parámetros.

En el Callback de conexión

```
"if (conectados==1) memcpy(&peerAddress,
param->connect.remote_bda, 6);
else if (conectados==2) memcpy(&peerAddress2,
param->connect.remote_bda, 6);
else if (conectados==3) memcpy(&peerAddress3,
param->connect.remote_bda, 6);"
```

Así logramos guardar las direcciones de cada dispositivo

```
-static void my_gap_event_handler(esp_gap_ble_cb_event_t
event, esp_ble_gap_cb_param_t* param) {
ESP_LOGW(LOG_TAG, "custom gap event handler, GAP
event: %d", (uint8_t)event);
rssI2 = param->read_rssi_cmpl.rssi;
```

Y así, llamando al evento `"esp_ble_gap_read_rssi(peerAddressX);"` con la dirección , podemos ver la RSSI del dispositivo conectado.

Esto, nos permite discernir entre varios dispositivos al mismo tiempo y sus medidas, pero tiene otro problema. Entre las medidas que se obtienen del RSSI, algunas de ellas nos dieron valores cercanos a -100db, es decir obtenemos medidas falsas. Por lo cual no pudimos seguir explorando este camino. Pueden haber varias razones de esas falsas medidas, entre ellas la velocidad con la cual las pedimos, algún retraso dentro de nuestro código o, su implementación no está del todo desarrollada por espresif. Este último comentario no es infundado, basta con explorar las soluciones dadas por la comunidad de desarrolladores para ver que este camino no ha sido explorado en su totalidad.

Dejamos expuesto esto, para mostrar el alcance de nuestro estudio sobre el ESP32 y como posible mejora a futuro. Si bien, a priori parecería que la solución por software (APP) seguiría siendo la mejor.

Por otro lado, la susceptibilidad de la medición del RSSI al ambiente, posición e interferencia es muy alta. Esto hace que ocurran aberraciones, que provocan por ejemplo, zonas en las que la medida es mejor que la pos anterior, pero en

realidad estamos más lejos del objetivo. Esto provoca que el robot se mueva alrededor de una zona incorrecta en loop. Nuestra solución consiste en agregar un movimiento diferente cada vez que el mapa se llena, generando “aleatoriedad”.

Nuestra solución fue implementar una antena externa más grande, pero por problemas de tiempo y la situación con el covid no logramos llevar a cabo este camino.

De haber sido así, la zona de cercanía hubiese sido más grande, permitiendo que los movimientos fueran menos pronunciados.

6.5.2. Necesidad de precisión

Para que nuestro proyecto funcione de manera exitosa se necesita que los movimientos sean lo más precisos posibles, esto es imposible dada la génesis de este proyecto, que involucra entre sus consignas la utilización de materiales reciclables. A medida que avanzó el proyecto fuimos utilizando varias maquetas, mejorando a prueba y error cada parte que genera imprecisiones. Podemos ver las distintas etapas de construcción en el anexo. Por ejemplo el giro, en nuestro caso particular es muy importante, por eso agregamos funciones para calibrar en ejecución de ser posible. No solo es un problema de construcción, materiales y motores en sí, si no que no existe un mecanismo de control que nos permita regular el giro de manera precisa.

Como extra , los sensores utilizados no siempre brindaron constancia, pero no afectan en gran efecto el desempeño del robot.

6.5.3. Estructura

Nuestra poca experiencia trabajando con estructuras, se vio reflejada en varios caminos erróneos y poca visión a futuro, consideramos que en esta parte logramos aprender mucho. El diseño final del robot, refleja nuestra experiencia durante el proceso, en esta última versión, es fácil acceder al interior del robot, para hacer cambios, cargar la batería y de así desecharlo conectar más sensores.

6.5.4. Desarrollo

Desarrollar de manera intensiva con el ESP32 fuera de las funciones “normales” es, como comentamos anteriormente, considerado difícil. Entender cómo se manejan las librerías oficiales y navegar entre la cantidad enorme de funciones que tienen, es en muchas veces, tedioso y sin mayores beneficios.

6.6. Conclusiones:

El ESP32 es un dispositivo increíble, compacto, lleno de funciones (Wi-Fi, bluetooth, dual core, GPIOs con múltiples funciones como touch etc) que además permite explorar varios campos de estudio. En este proyecto implementamos trabajo con sensores, motores, medidas de señal, programación concurrente, Arduino, app inventor, C++, animaciones, construcción, planificación e implementación de proyectos.

Desde el punto de vista de estudio, fue muy enriquecedor nos generó confianza sobre nuestras capacidades y nos alegra que, haya existido un ambiente donde se nos permitió crecer y elegir un proyecto que fuese de nuestro interés. Si bien insumo una cantidad de tiempo considerable (alrededor de 200 hs), nos enseñó lo que es llevar a cabo un proyecto desde su gestación.

Particularmente, sobre el desempeño del robot y los sensores de proximidad, consideramos que sus posibilidades son muy amplias. Por ejemplo, existen situaciones donde es mejor que ciertos sistemas respondan por proximidad, más aún considerando que el celular es hoy por hoy, el equivalente a un reloj años atrás. Es un dispositivo que siempre está con nosotros. Switches que además de ser inteligentes pueden reconocer la presencia humana, de una manera mucho más precisa que por ejemplo detección de movimiento. Por ejemplo automatizar el prendido y apagado de los monitores en zonas de trabajo. Así como las luces en espacios determinados.

Con elementos como seguridad, también se puede lograr hacer cerraduras que reconocen un celular, y abren. Sin la necesidad de abrir el celular, mostrar un código, o poner nuestra huella digital. No afirmamos que esta sea la mejor manera, pero es interesante pensar en las posibilidades.

6.7. Trabajo a Futuro:

Angle of Arrival, en Bluetooth 5.1, permite además saber el ángulo con el cual la señal llega desde otro dispositivo transmisor. Esto permite por mucho mejor posicionamiento y rastreo de dispositivos. Consideramos que este proyecto, realizado con esta tecnología arrojaría resultados muy interesantes, ya que en parte elimina gran parte del problema del algoritmo de búsqueda, y podría ayudar a evadir de mejor manera las aberraciones en las medidas.[15]

Por otro lado, si bien la MIT Inventor nos permitió crear una app de manera muy rápida, tiene sus deficiencias, por ejemplo no permite trabajar con procesos concurrentes. Esto nos trajo problemas en el momento de enviar el RSSI, ya que si enviamos demasiados datos tenemos que esperar a que llegue el turno de una nueva acción en el stack, y esto puede tardar

varios segundos. Por eso, generar una app con otro lenguaje puede también ser muy beneficioso para aprovechar mejor el procesamiento de otros dispositivos.

Por último la utilización de varios sensores o varios ESP32, también puede llegar a ser un objeto de estudio interesante, ya que si sumamos más elementos a la ecuación, podemos mejorar el algoritmo de búsqueda.

Muchos cambios de optimización se pueden lograr utilizando de mejor manera los semáforos e interrupciones, lo cual facilita la lectura del código y su posible reducción, así como mejorar su velocidad de respuesta.

7. Bitacora

- 30 de setiembre

Investigación e implementación de proximidad a través de BLE usando scan y RSSI

- 1 de octubre

Clase de consulta. Comentamos la dificultad de la comunicación BLE y la poca información disponible al respecto. Posible solución crear una app a través de MIT app inventor.

- 2 y 3 de octubre

2hs Investigación de app para esp32

- 4 de octubre

6hs Primer avance app MIT prender y apagar LED.

- 5 de octubre

11 horas MIT app esp 32 primeros avances

De aquí en adelante las modificaciones a la app son escasas.

- 9 de octubre

3hs de investigación(micropython)

- 10 de octubre

3 hs más de investigación(micropython)

- 15, 16, 17 de octubre

15 hrs funcionamiento de pantalla OLED y conectarse al ESP 32 a través de la app. Tuvimos problemas con las conexiones de la pantalla a la ESP32 debido a que por un lado el pinout de nuestra ESP32 era distinto al que estábamos aplicando y nos dimos cuenta debido a fallas en la pantalla cuando un sensor detectaba objetos tenía problemas que no eran comunes.

- 18 de octubre

3hs-Investigación pila de protocolos bluetooth, entender el comportamiento que se sigue para la comunicación entre dos objetos.

- **19 de octubre**

Investigación sensor HC-S04 e implementación en ESP32 8hrs

- **24/25/26 de octubre 15hs**

Investigación e implementación:

- multitasking
- deep sleep
- fc 51
- dual core
- Creación de bibliotecas para nuestras funciones

Las conclusiones que sacamos de los sensores es que no pueden ser utilizados como principales debido a que no cumplen con la expectativa. Al regular su potenciómetro para aumentar o disminuir la distancia de detección, el sensor comienza actuar de manera no deseada, y su detección por defecto ronda los 4 cm. Por lo que utilizaremos los sensores ultrasonido HC-SR04 para lo más delicado que es la parte de adelante.

- **27 de octubre**

Corrección problema con pantalla oled se apaga ante cambio de estado del led del esp32, se corrige y se aumenta a dos sensores al mismo tiempo modelo fc 51.

Investigación sobre motores hay dos candidatos que son motores reductores uno es 48:1 y el otro es 120:1 la diferencia es el torque de cada uno y que los 48:1 vienen con las ruedas de 6cm de diámetro y solo haríamos las ruedas traseras que acompañen el movimiento.

- **29 de octubre 9hs**

- Mover a la protoboard placa
- Core 0 es utilizada para las primera pruebas
- Instalamos ambos sensores
- Programamos el driver de los motores
- 1er Estructura de prueba para el robot
- Probar la comunicación de la app para movilizar al robot a distancia
- El robot se mueve
- Para la próxima traer mantel antideslizante para las ruedas
- Problemas al girar tal vez ocasionados por la distribución de peso

- **31 de octubre 9hs**

- Cambio de forma del robot

- Re soldar ruedas

- No pudimos implementar interrupciones ya que encendemos y apagamos los sensores, se optó por priorizar la tarea de lectura en los threads.

- Conexión a un transformador externo , baterías de 9 volts no aptas para periodo de prueba.

- Movimos la placa 1 al robot incluyendo los sensores.

- Primera implementación de un algoritmo autónomo para el robot.

- Problemas de consistencia con el rssi dificultan la tarea.

- El robot encuentra el target bastante bien.

- Los sensores son insuficientes (aún no tenemos el HC-SR04).

- **1ero de noviembre 2hs**

Investigación sobre antenas externas.

Modificaciones de arranque del sistema de búsqueda.

- **2 de noviembre**

Cambiamos la carcasa del auto

Modificaciones al código

Evaluación del peso en el giro

- **3 de noviembre (en clase)2:30**

Calibrar tiempo via bluetooth

Sensores

Las ruedas que se mueven

Ver el mapa

Resetear el mapa

Booster de voltaje./batería (Al final optamos por una batería 12V 1A)

- Antena, hablamos con Ilan la posibilidad de probar con una-seguir investigando

- El movimiento del robot tiene que depender de la señal RSSI.

- Controlar adelante atrás si pasa 2 veces seguidas reseteas el mapa ahí.

- **14 de noviembre 2hs**

Esperar Sensores

Inicio de documentación

Probar HC SR04

Probar Batería

Definir forma final

- **15-17 de noviembre**

- 3hs Sensores ,acomodamos el código.probamos colocar ota,corregimos algunas conexiones.

- Exploramos distintas maneras de hacer el multi conect Documentación.

- 21 de noviembre 9 Hrs

- Implementación de Robot en la estructura final.
- No se logra probar satisfactoriamente el robot, faltan detalles de estructura y detalles del código.
- Documentación

- 28-29 de noviembre 15 Hrs

- Solución de problemas con movimientos inesperados hacia atrás, era un error de código
- Además comprobamos retardos debido a problemas con Delay del código

- 1 de diciembre 12 Hrs

- Ultima versión de la estructura del robot, semáforos, mejoramiento de la APP, documentación, poster
- Seguimos reestructurando el formato final. por problemas con las ruedas.

- Continuamos con la documentación

- 5 de diciembre 11 Hrs

- Clase, arreglar giro, ruedas, mejorar el algoritmo de movimiento (GIROS, MARCHA ATRÁS, PARAR ANTES DE GIRAR, TIEMPOS, BAJAMOS EL ROZAMIENTO DE LAS RUEDAS, MEJORAMOS LOS LATERALES, Y VOLVIMOS A COLOCAR LA PANTALLA.

- 6 de diciembre 6 Hrs

- Mejora de algoritmos de distancia
- Implementación de animaciones para los estados
- Contador para modo deep sleep

9. Anexos

9.1. Bibliografía

1. BLE y fotos

<https://smart-lighting.es/bluetooth-low-energy-introduccion-la-tecnologia/#:~:text=El%20Bluetooth%20de%20baja%20energ%C3%ADa,dependientes%20de%20bater%C3%ADa%20o%20pila>

2. FC-51

http://www.dmf.unisalento.it/~denunzio/allow_listing/ARDUINO/FC51.pdf

3. Resistencia pull-up

9.2. Gestión del proyecto (Diagrama de Gantt)

A continuación se muestra primero el diagrama de Gantt inicial, con los tiempos estimados de las diferentes tareas. Luego, se muestra el diagrama de Gantt correspondiente a el tiempo de ejecución real de las mismas.

<http://www.learningaboutelectronics.com/Articles/LM393-comparator-circuit.php#:~:text=The%20LM393%20is%20dual,greater%20and%20which%20is%20smaller>

4. LM393

<http://www.learningaboutelectronics.com/Articles/LM393-comparator-circuit.php#:~:text=The%20LM393%20is%20dual,greater%20and%20which%20is%20smaller>

5. HC-SR04p

PDF de Elijah J. Morgan

6. Pantalla SSD1306 explicación

<https://www.youtube.com/watch?v=TAcVoh0xMRE>(robotics NV)

7. I2C vs SPI

<https://medium.com/@rjrajbir24/difference-between-i2c-and-spi-i2c-vs-spi-c6a68d7242c4>

8. PWM

<https://www.programoergosum.com/cursos-online/arduino/255-salidas-analogicas-pwm-con-arduino/salidas-analogicas-pwm#:~:text=Salidas%20PWM%20en%20Arduino,se%20env%C3%A1da%20una%20carga>

9. Puente H

[https://es.wikipedia.org/wiki/Puente_H_\(electr%C3%ADnica\)](https://es.wikipedia.org/wiki/Puente_H_(electr%C3%ADnica))

10. L298N

<https://www.prometec.net/l298n/>

11. Motorreductor

<https://saber.patagoniatec.com/2014/06/tt-gear-yellow-motor-motorreductor-amarillo-para-arduino-arduino-argentina-ptec/>

12. ESP 32

<https://saber.patagoniatec.com/2014/06/tt-gear-yellow-motor-motorreductor-amarillo-para-arduino-arduino-argentina-ptec/>

13. dmips

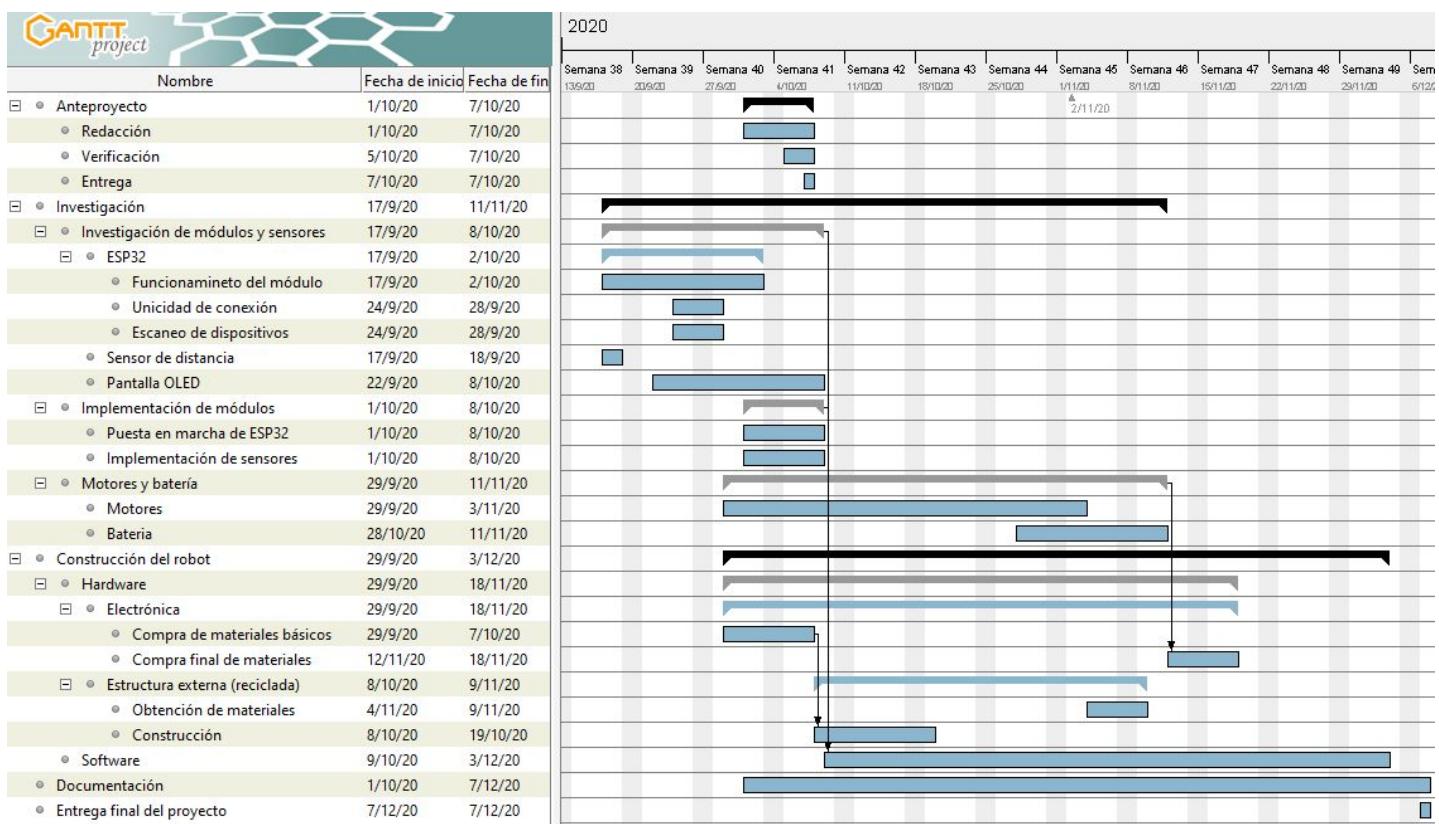
<https://raspberryparatorpes.net/glossary/dmips/>

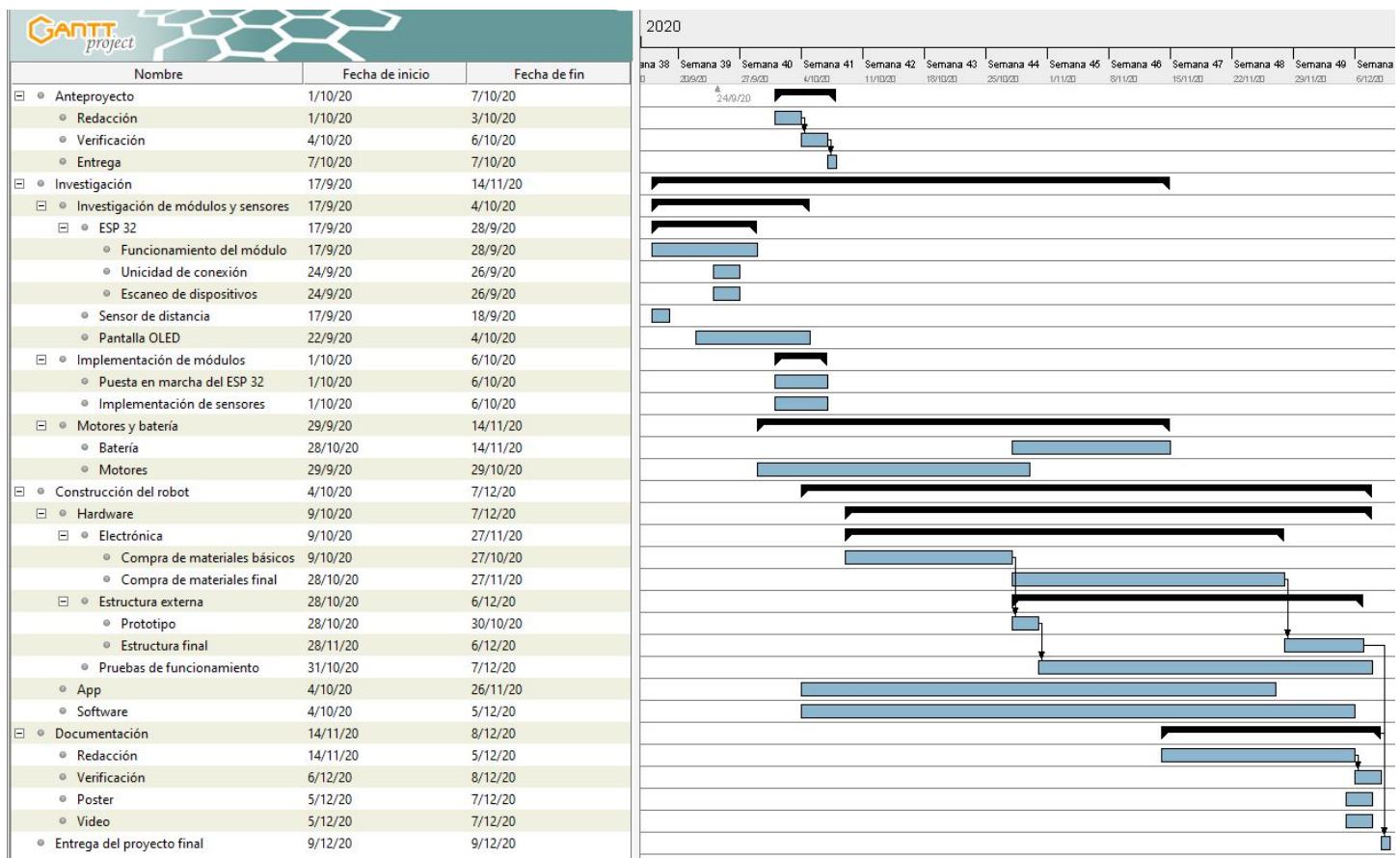
14. Dhystone

<https://es.wikipedia.org/wiki/Dhystone>

15. Bluetooth 5.1

<https://www.insightsip.com/news/what-s-new/471-bt5-1-direction-finding-and-positioning-with-ble>





9.3. Estructura del Robot

A continuación vemos las diferentes etapas de prueba y construcción por las que pasó nuestro robot.

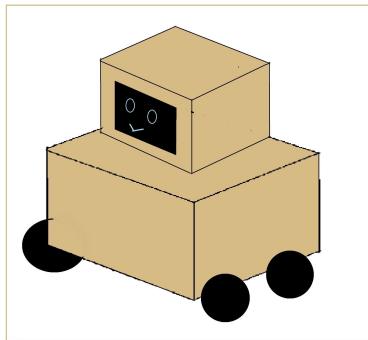
En la etapa 0, vemos el diseño original del robot, este fue una primera idea cuando todavía no sabíamos que componentes íbamos a utilizar ni cuál sería su posición final.

En la etapa 1, vemos el primer prototipo real de este proyecto. Su diseño es sencillo, ya que eran las primeras pruebas de movimiento y aún no descubrimos la forma óptima de colocar los motores y ruedas.

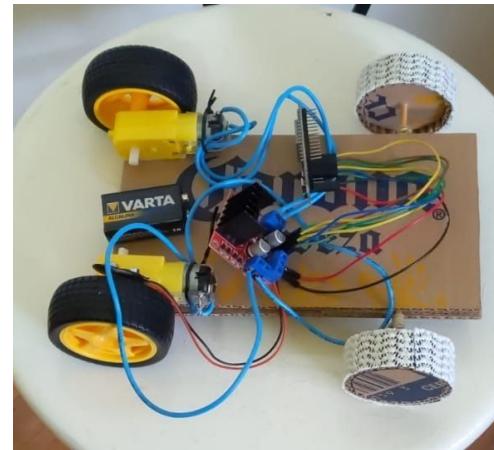
Debido a la sencillez de diseño del primer prototipo, el mismo no resistió un largo periodo de pruebas, lo que dio lugar a el segundo prototipo, etapa 2. El diseño se asemeja más a la versión final, en cuanto a posición de los componentes electrónicos y cantidad de sensores a utilizar.

En la tercera y última etapa, podemos ver el diseño final que se detalla a continuación.

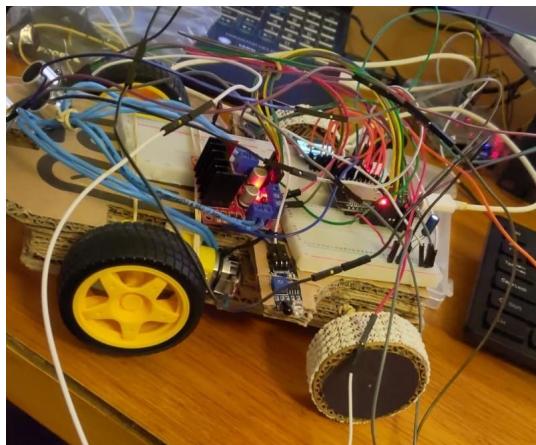
Etapa 0:



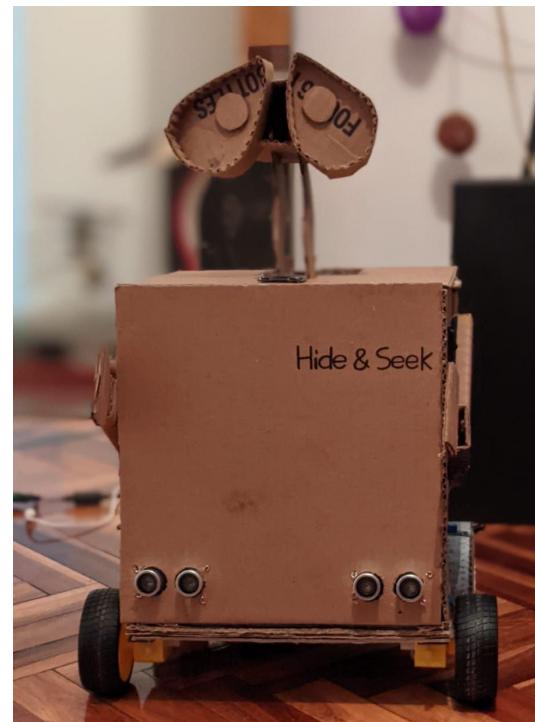
Etapa 1:



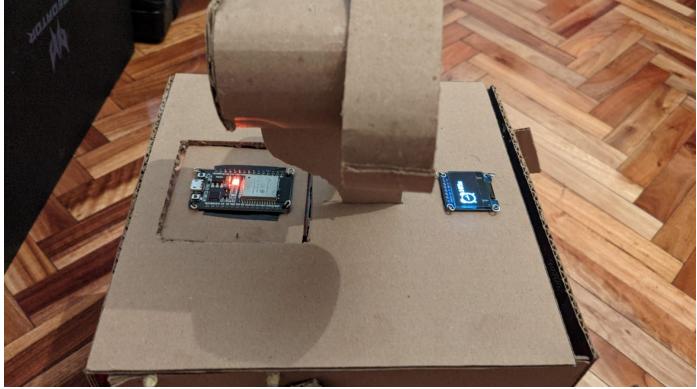
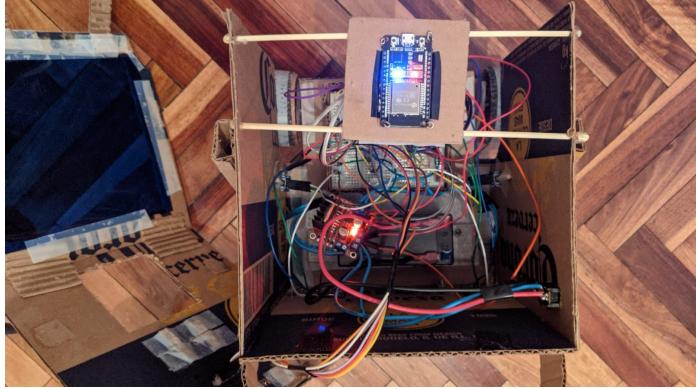
Etapa 2:



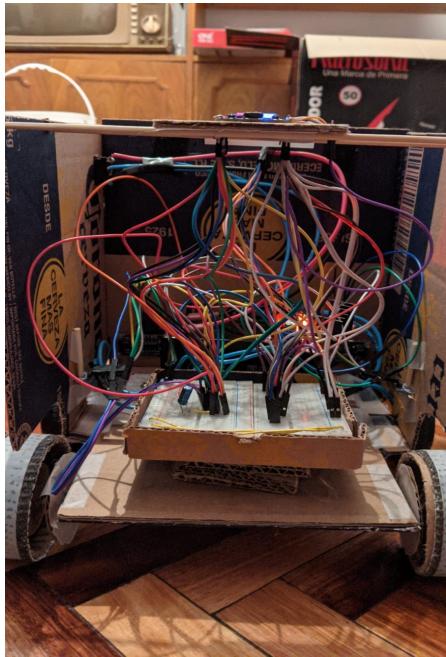
Etapa 3 (final):



Estructura final

Cara superior	Cara superior (expuesta)
	
Cara lateral	Cara frontal
	

Cara posterior (expuesta)



9.4. CÓDIGO

```
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEClient.h>
#include <BLEUtils.h>
#include <BLE2902.h>
#include <BLEScan.h>
#include <SPI.h>
#include <Wire.h>
#include "my_library.h"
#define pdTICKS_TO_MS( xTicks ) ( ( uint32_t )( xTicks ) * 1000 / configTICK_RATE_HZ )
#define SERVICE_UUID      "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
#define CHARACTERISTIC_UUID "beb5483e-36e1-4688-b7f5-ea07361b26a8"
#define SERVICE2_UUID      "45ecddcf-c316-488d-8558-3222e5cb9b3c"
/////////////////////////////Variable Globales relacionadas al ble
int vartiempo=20;
bool deviceConnected = false; //bandera de coneccion
int tiemporefresh = 1000;
```



```

int rssi = -48; //rssí medido entre esp32 y usuario
int rssi2 = -60;
int calibrar = -48; //distancia mínima que detecta el sensor
int constante = 2; //constante de ambiente
String valor = "";
String address = "";
esp_bd_addr_t peerAddress;// se guarda la address del cliente
esp_bd_addr_t peerAddress2;
esp_bd_addr_t peerAddress3;
int conectados=0;
//creacion de threads multitarea
TaskHandle_t Task1;//LOOP INFINITO EN CORE 0
TaskHandle_t Task2;//LOOP INFINITO EN CORE 1
TaskHandle_t Task3;//LOOP INFINITO EN CORE 0
//
int sensores[] = {HIGH, HIGH, HIGH, HIGH};
int mapa[] = {0, 0, 0, 0};//no tiene por que estar en el main
int LED_BUILTIN = 2;//LED AZUL
bool pica=false;//indica si te econtra
bool startbusqueda=false; //para pedirle al robot que arranque la busqueda

////// Touch pin
int threshold = 40;
bool touch2detected = false;
//////auxiliares
bool fowarden2=false;
bool iz=false;
bool der=false;
int tiempocalibrado=0;
TickType_t calibradostart;
//fin de juego
bool jugar=true;
//semaforo
SemaphoreHandle_t xMutex;

```

CALLBACKS

```
*****CALLBACKS*****  
*****
```

```
class MyCallbacks: public BLECharacteristicCallbacks { //ACA LLEGAN LOS COMANDOS VIA BLE
    void onWrite(BLECharacteristic *pCharacteristic) {
        std::string value = pCharacteristic->getValue();
        //Serial.print("core del callback");Serial.println(xPortGetCoreID());
```

```

if (value.length() > 0) {
    valor = "";
    String Salida = "";
    for (int i = 0; i < value.length(); i++) {
        // Serial.print(value[i]); // Presenta value.
        valor = valor + value[i];
        if (i > 0) {
            Salida = Salida + value[i];
        }
    }
    //Serial.println("*****");
}

if (valor.charAt(0) == 'r') {
    String smapa = "M";
    for (int i = 0; i < 4; i++) {
        smapa = smapa + mapa[i] + "-";
    }
    rssi = Salida.toInt();
    if (rssi>=-42) jugar=false;
    //Serial.println(rssi);
    std::string emapa((char*)&smapa, 9);
    pCharacteristic->setValue(emapa);
}
else if (valor.charAt(0) == '#') {

    Serial.println("CALIBRARTIEMPO");
    tiempocalibrado = pdTICKS_TO_MS(xTaskGetTickCount() - calibradostart);
    Serial.println(tiempocalibrado);
    pCharacteristic->setValue(tiempocalibrado);
}
else if (valor == "off") { //esto puede ser moverse adelante por ejemplo
    digitalWrite(LED_BUILTIN, LOW);
    Serial.println("LED turned on OFF");
    pCharacteristic->setValue("LED turned on OFF"); // Pone el numero aleatorio
}
else if (valor == "show") { //esto puede ser moverse adelante por ejemplo
    muestroTodo();
    Serial.println("Se cambio la bandera");
    pCharacteristic->setValue("bandera"); // Pone el numero aleatorio
}
else if (valor == "on") {
    digitalWrite(LED_BUILTIN, HIGH);
    Serial.println("LED turned on");
    pCharacteristic->setValue("LED turned on"); // Pone el numero aleatorio
}
else if (valor.charAt(0) == 'c') {
}

```

```

calibrar = Salida.toInt();
Serial.println("CALIBRADO");
Serial.println(calibrar);
Serial.println(Salida);
calibradostart = xTaskGetTickCount();
pCharacteristic->setValue("#");
}
else if (valor.charAt(0) == 'd') {
  pCharacteristic->setValue("duriendo");
  Dormir(threshold);
}
else if (valor.charAt(0) == 'a') {
  pCharacteristic->setValue("adelante");
  fowarden2= (fowarden2)? false :true;
}
else if (valor.charAt(0) == 'p') {
  pCharacteristic->setValue("parar");
  parar();
}
else if (valor.charAt(0) == 'i') {
  pCharacteristic->setValue("iz");
  iz=true;
  //moverIz();
}
else if (valor.charAt(0) == 'x') {
  pCharacteristic->setValue("derecha");
  der=true;
}
else if (valor.charAt(0) == 'b') {
  pCharacteristic->setValue("atras");
  moverAtras();
}
else if (valor == "s") {
  pCharacteristic->setValue("START");
  startbusqueda= (startbusqueda) ? false:true;
  pica=false;
}
else if (valor.charAt(0) == 't') {
  pCharacteristic->setValue("calibrogiro");
  tiempoDeGiro(Salida.toInt());
}
else if (valor.charAt(0) == 'v') {
  vartiempo=Salida.toInt();
}
else if (valor.charAt(0) == 'k') {

```

```

pCharacteristic->setValue("kalibroadelante");
constMotor(Salida.toInt());
}
else if (valor.charAt(0) == 'z') {
pCharacteristic->setValue("tiempo para pensar");
esperoExtra();
}
else if (valor.charAt(0) == '!') {
pCharacteristic->setValue("foward en el 2");
fowarden2= (fowarden2)? false :true;
}
else if (valor.charAt(0) == 'g') {
pCharacteristic->setValue("giroCambiado");
cambiarDer(Salida.toInt());
}
else if (valor.charAt(0) == 'e') {
pCharacteristic->setValue("cambio de eq");
elegirEq();
}
};

class myServerCallback : public BLEServerCallbacks {//ESTO ME DICE SI TENGO ALGUIEN CONECTADO

void onConnect(BLEServer* pServer, esp_ble_gatts_cb_param_t *param) {

char remoteAddress[18];
int aux = param->connect.conn_id;

//start sent the update connection parameters to the peer device.
sprintf(
remoteAddress,
"% .2X: % .2X: % .2X: % .2X: % .2X: % .2X",
param->connect.remote_bda[0],
param->connect.remote_bda[1],
param->connect.remote_bda[2],
param->connect.remote_bda[3],
param->connect.remote_bda[4],
param->connect.remote_bda[5]
);

ESP_LOGI(LOG_TAG, "myServerCallback onConnect, MAC: %s", remoteAddress);
Serial.print("onConnect, sos la conexion :");Serial.println(aux);
//Serial.println(aux);
}

```

```

conectados=conectados+1;
if (conectados==1) memcpy(&peerAddress, param->connect.remote_bda, 6);
else if (conectados==2) memcpy(&peerAddress2, param->connect.remote_bda, 6);
else if (conectados==3) memcpy(&peerAddress3, param->connect.remote_bda, 6);
//peerAddress=param->connect.remote_bda;
//rssI2 = esp_ble_gap_read_rssi(peerAddress);

//Serial.print("rssI durante la conexion"); Serial.println(rssI2);
//Serial.print("ADRESS durante la conexion"); Serial.println(remoteAddress);
esp_ble_tx_power_set(ESP_BLE_PWR_TYPE_CONN_HDL0, ESP_PWR_LVL_P9 );
esp_ble_tx_power_set(ESP_BLE_PWR_TYPE_CONN_HDL1, ESP_PWR_LVL_P9 );
deviceConnected = true;

}

void onDisconnect(BLEServer* pServer, esp_ble_gatts_cb_param_t *param) {

char remoteAddress[20];

sprintf(
remoteAddress,
"%02X:%02X:%02X:%02X:%02X",
param->disconnect.remote_bda[0],
param->disconnect.remote_bda[1],
param->disconnect.remote_bda[2],
param->disconnect.remote_bda[3],
param->disconnect.remote_bda[4],
param->disconnect.remote_bda[5]
);

ESP_LOGI(LOG_TAG, "myServerCallback onDisconnect, MAC: %s", remoteAddress);

deviceConnected = false;
conectados=conectados-1;
}
};

```

EVENTOS

```

*****EVENTOS*****
*****EVENTOS*****

static void my_gap_event_handler(esp_gap_ble_cb_event_t event, esp_ble_gap_cb_param_t* param) {
ESP_LOGW(LOG_TAG, "custom gap event handler, GAP event: %d", (uint8_t)event);
//Serial.print("RSSI status"); Serial.println(param->read_rssi_cmpl.status);
//Serial.print("RSSI del gap event");
rssI2 = param->read_rssi_cmpl.rssi;

```

```

Serial.print("Rssi medido en el evento:");Serial.println(rssi2);
//Serial.print("Address en gap "); Serial.println(BLEAddress(param->read_rssi_cmpl.remote_addr).toString().c_str());
}
  
```

```

//interrupciones
void gotTouch() { //se detecta interrupcion de tacto
  touch2detected = true;
}
  
```

Setup

```

*****SETUP*****
*****
*****BLEServer *pServer=NULL;
BLEAdvertising *pAdvertising=NULL;
void setup() {//SE INICIALIZA TODO
  Serial.begin(115200);
  iniciarMotores();
  initDisplay(1000);
  initSensorSonido();
  pinMode(LED_BUILTIN, OUTPUT);//onboard blue led
  ///////////BLE
  BLEDevice::init("Hide&Seek");// nombre del dispositivo bl
  BLEDevice::setPower(ESP_PWR_LVL_P9);
  esp_ble_tx_power_set(ESP_BLE_PWR_TYPE_DEFAULT, ESP_PWR_LVL_P9);
  esp_ble_tx_power_set(ESP_BLE_PWR_TYPE_CONN_HDL0,           ESP_PWR_LVL_P9
);esp_ble_tx_power_set(ESP_BLE_PWR_TYPE_CONN_HDL1, ESP_PWR_LVL_P9 );
  pServer = BLEDevice::createServer(); // Create the BLE Server
  BLEService *pService = pServer->createService(SERVICE_UUID);
  BLECharacteristic *pCharacteristic = pService->createCharacteristic(
    CHARACTERISTIC_UUID,
    BLECharacteristic::PROPERTY_READ |
    BLECharacteristic::PROPERTY_NOTIFY |
    BLECharacteristic::PROPERTY_WRITE
  );
  pServer->setCallbacks(new myServerCallback());
  pCharacteristic->setCallbacks(new MyCallbacks());
  BLEDevice::setCustomGapHandler(my_gap_event_handler);
  pCharacteristic->setValue("Iniciado.");
  pCharacteristic->addDescriptor(new BLE2902());
  esp_ble_tx_power_set(ESP_BLE_PWR_TYPE_ADV, ESP_PWR_LVL_P9);
  pService->start();
}
  
```

```

pAdvertising = pServer->getAdvertising();
pAdvertising->start();
///////////
touchAttachInterrupt(T3, gotTouch, 40); //SETEA LAS INTERRUPCION EN EL TOUCH 3
///////////////////////////////Task,multithreading
xMutex = xSemaphoreCreateMutex();

//create a task that will be executed in the Task2code() function, with priority 1 and executed on core 1
xTaskCreatePinnedToCore(
    Task2code, /* Task function. */
    "Task2", /* name of task. */
    10000, /* Stack size of task */
    NULL, /* parameter of the task */
    10, /* priority of the task */
    &Task2, /* Task handle to keep track of created task */
    1); /* pin task to core 1 */
delay(500);

xTaskCreatePinnedToCore(
    Task3code, /* Task function. */
    "Task3", /* name of task. */
    10000, /* Stack size of task */
    NULL, /* parameter of the task */
    1, /* priority of the task */
    &Task3, /* Task handle to keep track of created task */
    1); /* pin task to core 1 */

delay(500);
}
  
```

Main

```

*****MAIN
LOOPS*****
*****
```

```

void Task2code( void * pvParameters ) { // baja prioridad core 1
    TickType_t tiempodormir=xTaskGetTickCount();
    for (;;) { // infinite loop

        if(deviceConnected && !pica && startbusqueda && jugar){

            mapa[0] = 0; mapa[1] = 0; mapa[2] = 0; mapa[3] = 0;
```

```

cerebro(pica,rssi,rssi2,calibrar,mapa,sensores,jugar);

startbusqueda=false;//cerebro es un loop infinito salir de ahi implica haber cumplido las condiciones
tiempodormir=xTaskGetTickCount();
}else
{
  Serial.print("core del cerebro");Serial.println(xPortGetCoreID());
  imprimo(("Pica es: "), String(pica));
  if (fowarden2) {
    fowarden2=false;
    goFoward(10000,sensores,jugar);
    tiempodormir=xTaskGetTickCount();
  }else if (iz==true){
    moverIz();
    iz=false;
    tiempodormir=xTaskGetTickCount();
  }else if (der==true){
    moverDer();
    der=false;
    tiempodormir=xTaskGetTickCount();
  }
}

obstaculo(sensores);
xSemaphoreTake( xMutex, portMAX_DELAY );
if (!(estasBuscando() || estasMidiendo())) reloj(int(pdTICKS_TO_MS(xTaskGetTickCount() - tiempodormir)/1000));
xSemaphoreGive( xMutex );
jugar=true;

if (pdTICKS_TO_MS(xTaskGetTickCount() - tiempodormir)>240000) Dormir(threshold);

}

void Task3code( void * pvParameters ) {
  for (;;) { // infinite loop
    //Serial.println("estoy al pedo");
    xSemaphoreTake( xMutex, portMAX_DELAY );
    if (estasBuscando()) lupa(rssi);
    else if (estasMidiendo()) phone(rssi);
    xSemaphoreGive( xMutex );
    cuentaOvejas(touch2detected, threshold);
    vTaskDelay(200 / portTICK_PERIOD_MS);
  }
}

```

```
void loop() {//por default el loop va al core 1
```

```
}
```

9.4.1. Libreria

```
#include "my_library.h"
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

//Display
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
// Declaration for SSD1306 display connected using software SPI (default case):
#define OLED_MOSI 23
#define OLED_CLK 18
#define OLED_DC 4 //4
#define OLED_CS 5
#define OLED_RESET 17
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, OLED_MOSI, OLED_CLK, OLED_DC,
OLED_RESET, OLED_CS);

bool mostrarprints = false;

///////////Variable para los motores/L298N
// Motor B//iz mirando de atras
int motorBpin1 = 22;//27
int motorBpin2 = 26;//26
int enableBpin = 16;//14

// Motor A//der mirando de atras
int motorApin1 = 33; //Pin1 del driver
int motorApin2 = 32; //Pin2 del driver
int enableApin = 25;

// Setting PWM properties
const int freq = 30000;
const int pwmChannelA = 0;
const int pwmChannelB = 1;
const int resolution = 8;
int dutyCycle = 140;

///////////variables para

int sensorPinIz = 35; //el out del sensor esta en el pin 34
int sensorPinPowerIz = 19; //el vcc del sensor esta en el pin 22
```

```
//int sensorPinAde = 13 ;//el out del sensor esta en el pin 35
//int sensorPinPowerAde = 12 ;//el out del sensor esta en el pin 35
int sensorPinDer = 34;//el out del sensor esta en el pin 35
int sensorPinPowerDer = 21;//el power del sensor esta en el pin 21
int sensorValue = HIGH; //high es no , me parece poco saludable
//sonido
// variables del los HC-sr04
#define echoPin 27 // attach pin D2 Arduino to pin Echo of HC-SR04 19
#define trigPin 14 //attach pin D3 Arduino to pin Trig of HC-SR04
#define echoPin2 12
#define trigPin2 13
long duration = 0; // variable for the duration of sound wave travel
int distance = 0; // variable for the distance measurement
long duration2 = 0;
int distance2 = 0;
*****bandera global
bool obstacle = false;// evita que se active parar varias veces, indica que hay alguien delante
```

SENSORES

```
////////////////////////////Funciones RELACIONADAS A LOS SENSORES //////////////////////\\
//*****\\
```

```
int leerSensor(int pinsensor, int powerPinSensor) {
    pinMode(powerPinSensor, OUTPUT); //gpio 22 es output
    digitalWrite(powerPinSensor, HIGH); //gpio 22 es high,1,3.3v
    delay(20); // no gusta
    //esperoMillis(20);
    pinMode(pinsensor, INPUT); //este esta conectado al out del sensor
    int salida = digitalRead(pinsensor);
    digitalWrite(powerPinSensor, LOW); //gpio 22 es high,1,3.3v

    return salida;
}
void initSensorSonido() {
    pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
    pinMode(trigPin2, OUTPUT); // Sets the trigPin as an OUTPUT
    pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
    pinMode(echoPin2, INPUT); // Sets the echoPin as an INPUT
    //Serial.println("Ultrasonic Sensor HC-SR04 Test"); // print some text in Serial Monitor
    //Serial.println("with Arduino UNO R3");
}
int leerSensorSonido() {
    int salida = HIGH;
    // Clears the trigPin condition
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
```

```

digitalWrite(trigPin, LOW);
// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH);

digitalWrite(trigPin2, LOW);
delayMicroseconds(2);
// Sets the trigPin HIGH (ACTIVE) for 10 microseconds
digitalWrite(trigPin2, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin2, LOW);
duration2 = pulseIn(echoPin2, HIGH);

// Calculating the distance
distance = duration * 0.034 / 2;      // Speed of sound wave divided by 2 (go and back)
distance2 = duration2 * 0.034 / 2;    // Speed of sound wave divided by 2 (go and back)
// Displays the distance on the Serial Monitor

if ((distance < 22 && distance > 0) || (distance2 < 22 && distance2 > 0)) {
  salida = LOW;
} else salida = HIGH;

if (salida == LOW && !obstacle) {    //esto hace que no pares todo el tiempoo
  parar();
  obstacle = true;
} else if (salida == HIGH) {
  obstacle = false;
}
(salida == LOW ) ? digitalWrite(2, HIGH) : digitalWrite(2, LOW);
imprimo("Distance 1: ", String(distance));
imprimo("Distance 2: ", String(distance2));
return salida;
}

void leerSensores(int sensors[]) {
  sensors[0] = leerSensor(sensorPinIz, sensorPinPowerIz);      //izqueirda
  sensors[1] = leerSensor(sensorPinDer, sensorPinPowerDer);    //derecha
  sensors[2] = leerSensorSonido(); //adelante
  // sensors[3] = leerSensor(sensorPinBack,sensorPinPowerBack);
}

int obstaculo(int sensors[]) {
  /////////////////////////////////Sensores
  leerSensores(sensors);
  int sensorValueIz = sensors[0];
  int sensorValueDer = sensors[1];
  int sensorValueAde = sensors[2];
  int sensorValueBack = HIGH; //esto es para probar nomas
  //int sensorValueDer = sensores[1];
  // int sensorValueAde = sensores[2];
  //int sensorValueBack = sensores[3];
  int salida = 0;
  if (sensorValueIz == LOW) {
    salida = 1;
  }
}

```

```

}

if(sensorValueDer == LOW) {
  salida = salida + 2;
}
if(sensorValueAde == LOW) {
  salida = salida + 7;
}
if(sensorValueBack == LOW) {
  salida = salida + 11;
}
(salida > 0) ? digitalWrite(2, HIGH) : digitalWrite(2, LOW);
(salida > 0) ? Serial.println("obstaculo") : Serial.println("");
switch (salida) {
  case 1: imprimo("!!, !! Izquierda", ""); break;
  case 2: imprimo("!!, !! Derecha", ""); break;
  case 3: imprimo("!!, !! Derecha e Izquierda", ""); break;
  case 7: imprimo("!!, !! Adelante", ""); break;
  case 8: imprimo("!!, !! Adelante e Izquierda", ""); break;
  case 9: imprimo("!!, !! Adelante y Derecha", ""); break;
  case 10: imprimo("!!, !! Adelante , Izquierda y Derecha", ""); break;
  case 11: imprimo("!!, !! Atras", ""); break;
  case 12: imprimo("!!, !! Atras e Izquierda", ""); break;
  case 13: imprimo("!!, !! Atras e Derecha", ""); break;
  case 14: imprimo("!!, !! Atras , Izquierda y Derecha", ""); break;
  case 18: imprimo("!!, !! Atras y Adelante", ""); break;
  case 19: imprimo("!!, !! Atras , Adelante Izquierda", ""); break;
  case 20: imprimo("!!, !! Atras , Adelante y Derecha", ""); break;
  case 21: imprimo("ATASCADO COMPLETAMENTE", ""); break;
  default : Serial.println("clear"); break;
}
return (salida);
}

```

FUNCIONES EXTRAS

```

*****FUNCION EXTRAS*****
*****funcion distancia*****
double distancia(int rssia, int calibrara, int constantea) {
  double dis ;
  double pot = (calibrara - rssia) / (10 * constantea);
  dis = pow(10, dis) - 1;
  return dis;
}
int listaCircular(int rssi[], int largo) {
  int promedio = rssi[0];
  for (int i = 1; i < largo; i++) {
    promedio = rssi[i] + promedio;
  }
}
///////////
void imprimo(String mensaje1, String mensaje2) {

```

```

if (mostrarprints) {
  Serial.print(mensaje1); Serial.println(mensaje2);
}
}
void muestroTodo() {
  if (mostrarprints) {
    mostrarprints = false;
  } else mostrarprints = true;
}

///////////////////////////////
#define pdTICKS_TO_MS( xTicks ) ( ( uint32_t )( xTicks ) * 1000 / configTICK_RATE_HZ )
static void esperarMillis(double espero )
{
  int tiempo = 0;
  TickType_t time_start = xTaskGetTickCount();
  while (1) {
    tiempo = pdTICKS_TO_MS(xTaskGetTickCount() - time_start);
    if ( tiempo > espero ) {
      Serial.print("Millis transcurridos :"); Serial.println(espero);
      break;
    }
    vTaskDelay(10 / portTICK_PERIOD_MS);
  }
}
  
```

FUNCIONES Display

```

*****FUNCIONES Display*****
*****FUNCIONES Display*****
void initDisplay(int time) { //inicializa el display
  // SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3D)) {
    Serial.println(F("SSD1306 allocation failed"));
    for (;;) // Don't proceed, loop forever
  }
  display.clearDisplay();
  display.drawBitmap(0, 0, Cara5, 128, 64, WHITE);
  display.display();
  delay(time);
  display.drawBitmap(0, 0, Cara5, 128, 64, WHITE);
  display.invertDisplay(true);
  delay(time);
  display.invertDisplay(false);
  delay(time);
  display.clearDisplay();
  reloj(0);
}
void testdrawstyle(String valor, int rssi, int calibrar, int constante) { //muestra valores
  display.clearDisplay();
  
```

```

display.setTextSize(2);           // Normal 1:1 pixel scale
display.setTextColor(SSD1306_WHITE); // Draw white text
display.setCursor(0, 0);         // Start at top-left corner
//display.println(F("Te voy a encontrar!"));

display.setTextColor(SSD1306_BLACK, SSD1306_WHITE); // Draw 'inverse' text
display.println(valor);

display.setTextSize(2);           // Draw 2X-scale text
display.setTextColor(SSD1306_WHITE);
double a = distancia(rssi, calibrar, constante);
display.print(a);
display.println(F("m-- "));
display.print(F("-Rssi-> "));
display.print(rssi);
display.print(F("-TX0-> "));
display.print(calibrar);

//display.print(F("--TX0" ));

display.display();
//delay(1000);
}

///////////
void Display(bool conectado, String valori, int rssii, int calibrari, int constantei) {

if (conectado) {
  testdrawstyle(valori, rssii, calibrari, constantei);
}
else {
  mostrarImagen(Cara1, 1000, true, "");
  mostrarImagen(Cara2, 1000, true, "");

}
bool buscando=false;
bool estasBuscando(){
  return buscando;
}

```

FUNCIONES IMAGEN

```

*****FUNCIONES IMAGEN*****
*****funcionesImagen*****

void mostrarTexto(String texto) {
  display.setCursor(50, 10);      // Start at top-left corner
  display.setTextColor(SSD1306_WHITE); // Draw white text
  display.setTextSize(2);        // Draw 2X-scale text
  display.setTextColor(SSD1306_WHITE);
  display.print(texto[0]);display.print(texto[1]);display.print(texto[2]); display.println(F("s" ));

```

```

}

void mostrarTextoRssiAnterior(String texto) {
    display.setCursor(60, 30);          // Start at top-left corner
    display.setTextColor(SSD1306_WHITE); // Draw white text
    display.setTextSize(2);           // Draw 2X-scale text
    display.setTextColor(SSD1306_WHITE);
    display.print(F("Ro:"));display.print(texto[0]);display.print(texto[1]);display.println(texto[2]);
}

void mostrarTextoRssiMedido(String texto) {
    display.setCursor(60, 50);          // Start at top-left corner
    display.setTextColor(SSD1306_WHITE); // Draw white text
    display.setTextSize(2);           // Draw 2X-scale text
    display.setTextColor(SSD1306_WHITE);
    display.print(F("Rp:"));display.print(texto[0]);display.print(texto[1]);display.println(texto[2]);
}

void mostrarImagen(const unsigned char imagen [] PROGMEM, int tiempo, bool full, String texto) {
    display.clearDisplay();
    if (full) display.drawBitmap(0, 0, imagen, 128, 64, WHITE);
    else {
        display.drawBitmap(0, 8, imagen, 48, 48, WHITE);
    }
    mostrarTexto(texto);
    display.display();
    vTaskDelay(tiempo / portTICK_PERIOD_MS);
}

void reloj(double segundos) {
    mostrarImagen(frame0, 37, false, String(segundos));
    mostrarImagen(frame1, 37, false, String(segundos)); mostrarImagen(frame2, 37, false, String(segundos));
    mostrarImagen(frame3, 37, false, String(segundos)); mostrarImagen(frame4, 37, false, String(segundos));
    mostrarImagen(frame5, 37, false, String(segundos)); mostrarImagen(frame6, 37, false, String(segundos));
    mostrarImagen(frame7, 37, false, String(segundos)); mostrarImagen(frame8, 37, false, String(segundos));
    mostrarImagen(frame9, 37, false, String(segundos)); mostrarImagen(frame10, 37, false, String(segundos));
    mostrarImagen(frame11, 37, false, String(segundos)); mostrarImagen(frame12, 37, false, String(segundos));
    mostrarImagen(frame13, 37, false, String(segundos)); mostrarImagen(frame14, 37, false, String(segundos));
    mostrarImagen(frame15, 37, false, String(segundos)); mostrarImagen(frame16, 37, false, String(segundos));
    mostrarImagen(frame17, 37, false, String(segundos)); mostrarImagen(frame18, 37, false, String(segundos));
    mostrarImagen(frame19, 37, false, String(segundos)); mostrarImagen(frame20, 37, false, String(segundos));
    mostrarImagen(frame21, 37, false, String(segundos)); mostrarImagen(frame22, 37, false, String(segundos));
    mostrarImagen(frame23, 37, false, String(segundos)); mostrarImagen(frame24, 37, false, String(segundos));
    mostrarImagen(frame25, 37, false, String(segundos)); mostrarImagen(frame26, 37, false, String(segundos));
    mostrarImagen(frame27, 37, false, String(segundos));
}

void phone(double segundos) {
    mostrarImagen(phone0, 37, false, String(segundos));
}

```


}

FUNCIONES Deep Sleep

```
*****FUNCIONES Deep Sleep*****
void callbackDormir() { //esto es por si nescessitamos quese hagan cosas antes de ir a dormir
  imprimo("entro al callback"), "";
}
void Dormir(int threshold) {
  /* Greater the value, more the sensitivity */
  touch_pad_t touchPin;
  touchAttachInterrupt(T3, callbackDormir, threshold);

  //Configure Touchpad as wakeup source
  esp_sleep_enable_touchpad_wakeup();
  //Go to sleep now
  Serial.println("A dormir");
  esp_deep_sleep_start();
}

void cuentaOvejas(bool &touch2detected, int threshold) {

if (touch2detected) {
  touch2detected = false;
  int count = 0;
  bool sleep = false;
  Serial.println("en el cuentaovejas");
  while (touchRead(T3) <= threshold) {
    //Serial.println(touchRead(T3));
    count = count + 1;
    if (count == 5000) {
      imprimo(String(count / 1000), "-");
      delay(5000); //mejor mostrar animacion de 5000s
      Dormir(threshold);
    }
    (count % 1000 == 0) ? Serial.print(count / 1000) : false;
  }
  imprimo("Toque detectado"), "";
}
}
```

FUNCIONES MOTOR

```
*****FUNCIONES motor*****
```

```
*****void iniciarMotores() {
  pinMode(motorApin1, OUTPUT);
  pinMode(motorApin2, OUTPUT);
  pinMode(enableApin, OUTPUT);
  pinMode(motorBpin1, OUTPUT);
  pinMode(motorBpin2, OUTPUT);
  pinMode(enableBpin, OUTPUT);
  // configure LED PWM functionalites
  ledcSetup(pwmChannelA, freq, resolution);
  ledcSetup(pwmChannelB, freq, resolution);
  ledcAttachPin(enableApin, pwmChannelA);
  ledcAttachPin(enableBpin, pwmChannelB);

  // attach the channel to the GPIO to be controlled

  digitalWrite(motorApin1, LOW);
  digitalWrite(motorApin2, LOW);
  digitalWrite(motorBpin1, LOW);
  digitalWrite(motorBpin2, LOW);
}
```

FUNCIONES MOVIMIENTO

```
*****FUNCTION MOVIMIENTO*****
*****FUNCTION MOVIMIENTO*****
*****FUNCTION MOVIMIENTO*****

void moverAtras() {
  dutyCycle = 180;
  //if (dutyCycle > 200) dutyCycle = 200;
  Serial.print("Back with duty cycle: ");
  Serial.println(dutyCycle);
  ledcWrite(pwmChannelA, dutyCycle);
  ledcWrite(pwmChannelB, dutyCycle);
  digitalWrite(motorApin1, LOW);
  digitalWrite(motorApin2, HIGH);
  digitalWrite(motorBpin1, LOW);
  digitalWrite(motorBpin2, HIGH);
  ledcWrite(pwmChannelA, dutyCycle);
  ledcWrite(pwmChannelB, dutyCycle);
}

void parar() {

  digitalWrite(motorApin1, LOW);
  digitalWrite(motorApin2, LOW);
  digitalWrite(motorBpin1, LOW);
  digitalWrite(motorBpin2, LOW);
  Serial.println("Parar: ");
  dutyCycle = 140;
}

int motorcali = 15;
void constMotor(unsigned int a) {
  motorcali = a;
```

```

}

void moverAdelante() {
  dutyCycle = 180;
  //if (dutyCycle > 200) dutyCycle = 200;
  if (!obstacle) {
    ledcWrite(pwmChannelA, dutyCycle + motorcali);
    ledcWrite(pwmChannelB, dutyCycle);
    digitalWrite(motorApin1, HIGH);
    digitalWrite(motorApin2, LOW);
    digitalWrite(motorBpin1, HIGH);
    digitalWrite(motorBpin2, LOW);
    ledcWrite(pwmChannelA, dutyCycle);
    ledcWrite(pwmChannelB, dutyCycle);
    //Serial.print("Se mueve adelante: "); Serial.println(dutyCycle+motorcali);
  }
}

int tiempogiro = 550;
void tiempoDeGiro(int tg) {
  tiempogiro = tg;
}
int tiempogiroiz = 550;
void tiempoDeGiroIZ(int tg) {
  tiempogiroiz = tg;
}
void moverIz() {
  dutyCycle = 180;
  //if (dutyCycle > 255) dutyCycle = 255;
  Serial.print("Giro iz "); Serial.println(dutyCycle);
  ledcWrite(pwmChannelA, dutyCycle);
  ledcWrite(pwmChannelB, dutyCycle);
  digitalWrite(motorApin1, LOW);
  digitalWrite(motorApin2, HIGH);
  digitalWrite(motorBpin1, HIGH);
  digitalWrite(motorBpin2, LOW);
  ledcWrite(pwmChannelA, dutyCycle);
  ledcWrite(pwmChannelB, dutyCycle);
  esperarMillis(tiempogiro);

  parar();
}
int constDer = 15;
void cambiarDer(int cons) {

  constDer = cons;
}
void moverDer() {
  dutyCycle = 180;
  //if (dutyCycle > 255) dutyCycle = 255;
  Serial.print("Giro derecha: "); Serial.println(dutyCycle);
  ledcWrite(pwmChannelA, dutyCycle + constDer);
}

```

```

ledcWrite(pwmChannelB, dutyCycle);
digitalWrite(motorApin1, HIGH);
digitalWrite(motorApin2, LOW);
digitalWrite(motorBpin1, LOW);
digitalWrite(motorBpin2, HIGH);
ledcWrite(pwmChannelA, dutyCycle);
ledcWrite(pwmChannelB, dutyCycle);
esperarMillis(tiempogiro);
//esperoMillis(tiempogiro);
parar();
}
  
```

FUNCIONES DE BÚSQUEDA

```

***** FUNCIONES ***** DE *****
*BÚSQUEDA*****
***** *****
***** marcarMapa(int mapa[]) {
    mapa[0] = 1;
}
***** girarMapa(int mapa[], int dir) { //if dir ==0 entonces iz
    if (dir == 0) {
        mapa[3] = mapa[2];
        mapa[2] = mapa[1];
        mapa[1] = mapa[0];
    }
    mapa[0] = 0;
}

***** decidir(int mapa[], int sensores[]) {
    bool giroIz = true;      //libre el sensor
    bool giroDer = true;     //libre el sensor
    bool reseteo=false;
    //chequeo los sensores para ver mis opciones
    if (sensores[0] == 0) { //tenes a alguien a la iz
        giroIz = false;
    }
    else if (sensores[1] == 0) { //tenes a alguien a la der
        giroDer = false;
    }

    ///////////mapa
    if ((mapa[0] + mapa[1] + mapa[2] + mapa[3]) == 4 ) { //implica que el mapa esta lleno
        reseteo=true;
    }
}
  
```

```

mapa[3] = 0;
mapa[2] = 0;
mapa[1] = 0;
mapa[0] = 0;
}
if (mapa[1] + mapa[3] == 2)    //giro 180 y no me importa nada
  Serial.println("giro 180");
  girarMapa(mapa, 0);
  moverIz();
  girarMapa(mapa, 0);
  moverIz();
}
else if (mapa[3] == 0 && giroIz) { // me muevo a la iz
  Serial.println("giro izquierda");
  girarMapa(mapa, 0);
  moverIz();
}
// me muevo der
else if (mapa[1] == 0 && giroDer) {
  Serial.println("giro derecha");
  girarMapa(mapa, 1);
  moverDer();
}
if (reseteo) goFoward(1000, sensores,reseteo);
}
///////////
bool estoymidiendo=false;
bool estasMidiendo(){
  return estoymidiendo;
}
int medirRssi( int &rssi, long tiempo) {
  //unsigned long time;
  //time = millis();
  estoymidiendo=true;
  int aux = 0;
  int count = 0;
  int* arrayrssi = new int[60];
  TickType_t time_start = xTaskGetTickCount(); // toma el tiempo actual
  int trecorrido = 0;
  while (trecorrido < tiempo) {
    trecorrido = pdTICKS_TO_MS(xTaskGetTickCount() - time_start);
    arrayrssi[count] = rssi;
    count = count + 1;
    aux = (rssи + aux);
    vTaskDelay(150 / portTICK_PERIOD_MS);
  }
  aux = aux / count;
  //varianza
  double varianza = 0;
  for (int i = 0; i < count; i++) {
    varianza = varianza + (arrayrssi[i] - aux) * (arrayrssi[i] - aux);
  }
}

```

```

varianza = varianza / count;
sqrt(varianza);
delete[] arrayssi;
imprimo(("Varianza: "), String(varianza));
imprimo(("Cantidad de veces que promedio: "), String(count));
imprimo(("Promedio de rssi: "), String(aux));
Serial.print("Promedio de rssi: "); Serial.println( String(aux));
if (varianza>10) { return medirRssi( rssi, tiempo); }
//else
estoymidiendo=false;
return aux;
}
///////////
bool eq1 = false;
void elegirEq() {
eq1 = (eq1) ? false : true;
}
double distanciaRssi(int rssimedido) {
double salida = 0;
double rssid = abs(-rssimedido);
if (rssid <= 40) {
salida = 501; //si ocurre alguna averracion no se cae
}
else if (eq1) salida = (-1 / 300) * pow((rssid), 2) - (1 / 2) * rssid - 85 / 6 ;
else salida = (-0.00179) * pow((rssid), 2) + (0.238) * rssid - (246) / rssid ;
//salida = pow((rssid / 100 + 0.5), 2) - 1 / rssid - 0.4;s
//a*(-80)^2-b*(80)+c=4.5,a*(-50)^2-b*(50)+c=2.5,a*(-40)^2-b*(40)+c=0.5
Serial.print("Se calcula una distancia de:"); Serial.println(salida);
return salida * 1000;
}
///////////
double goFoward(double distancia, int sensores[],bool &jugar) {
buscando=true;
leerSensorSonido();
TickType_t time_start = xTaskGetTickCount(); // toma el tiempo actual
double recorrido = 0;
while (!obstacle&& jugar) {
recorrido = pdTICKS_TO_MS(xTaskGetTickCount() - time_start); //comparo con el tiempo inicial
leerSensorSonido(); // leo sensor
moverAdelante(); // me muevo , depende de la bandera obstacle
if (obstacle) {//tengo alguien delante?
Serial.print("Obstaculo=Millis transcurridos :"); Serial.println(recorrido);
buscando=false;
parar();
}
return recorrido;
}
if ( recorrido > distancia) {
Serial.print("Millis transcurridos :"); Serial.println(recorrido);
break;
}
}

```

```

    vTaskDelay(10 / portTICK_PERIOD_MS);
}
parar();
leerSensores(sensores);//le todos los sensores
buscando=false;
return recorrido;
}
///////////
bool esperarextra = false;
bool esperoExtra() {
  esperarextra = (esperarextra) ? false : true;
}
  
```

CEREBRO

```

*****CEREBRO*****
*****
*****CEREBRO*****
*****
int rssicero=0;
int getRssiAnterior(){
  return rssicero;
}
void setRssiAnterior(int rssip){
  rssicero=rssip;
}
void cerebro(bool &bandera, int &rssi, int &rssi2, int &calibrar, int mapa[], int sensores[],bool &jugar) {
  int rssianterior = rssi;
  int rssipromedio = medirRssi(rssi, 6000);
  int distancia = distanciaRssi(rssipromedio); //es el tiempo de recorrdia
  while (rssi < calibrar && jugar) {
    imprimo("Mapa antes de arrancar a moverse", "");
    distancia = goFoward(distancia, sensores,jugar); //devuelve cuanto se movio
    if (!jugar) break;
    rssipromedio = medirRssi(rssi, 6000);
    //setRssiAnterior(rssipromedio);
    if ((rssipromedio < rssianterior ) {      //empeora, me muevo para atras
      marcarMapa(mapa);
      moverAtras();
      esperarMillis(distancia);           //te moves lo mismo que avanzaste
      parar();
      //vTaskDelay(100 / portTICK_PERIOD_MS);
      //esperarMillis(2000);
      if (!jugar) break;
      calculadora(rssipromedio);
      calculadora(rssipromedio);
      decidir(mapa, sensores);
    }
    else if (obstacle) {                //la señal es mejor pero tengo un obstaculo
      moverAtras();
    }
  }
}
  
```

```

moverAtras();
esperarMillis(500);
parar();
if (!jugar) break;
calculadora(rssipromedio);
calculadora(rssipromedio);
decidir(mapa, sensores);
}
rssianterior = rssipromedio;
distancia = distanciaRssi(rssipromedio); //determino cuanto me voy a mover
if (esperarextra) {
  imprimo(("tiempo extra"), "");
  esperarMillis(5000);
}
}
Serial.println("te encontre y me canse");
bandera = true;
}

```

9.4.2. HEADERS

```

#ifndef MY_LIBRARY_H
#define MY_LIBRARY_H
#include <Arduino.h>

int getRssiAnterior(); //Devuelve el rssi de la pos atnerior
void setRssiAnterior(int rssip); //
void lupa(double segundos);
bool estasMidiendo();//bandera de animacion
bool estasBuscando();//bandera de animacion
void phone(double segundos);
void mostrarTexto(String texto);
void cambiarDer(int cons);//para calibrar el giro derecho;
void initOta();//inicializa Over the air
void conexionOta();// mantiene la conexion
void imprimo(String mensaje1, String mensaje2); //imprime en pantalla depende de bandera global
void muestroTodo(); //cambia la bandera global
void testdrawstyle(String valor, int rssi, int calibrar, int constante); // Muestra los valroes pasados por referencia
int obstaculo(int sensors[]); //devuelve y muestra que sensores se estan utilizando, 0 en caso de clear path
double distancia(int rssia, int calibrara, int constantea); //transforma rssi a distancia
void Display(bool conectado, String valor, int rssii, int calibrari, int constantei); // Muestra los valores de rssi o la cara del
robot
void initDisplay(int time); //inicializa el display
void animacion(String animacion, int final); //muestra una animacion en el display
void Dormir(int threshold); //Pone en DeepSleep al sistema
void callbackDormir(); //es una funcion que se activa antes de poner el sistema a dormir
void gotTouch2(); //cambia la bandera en interrupciones de tacto
void cuentaOvejas(bool &touch2detected, int threshold); //verifica que el touch pin se toque por 5 segundos de ser asi entra a
Dormir
int leerSensor(int pinsensor, int powerPinSensor); //Prende el sensor, lee su valor y lo apaga.
void initSensorSonido(); //inicializa los sensores de sonido

```



```

0xff, 0xff
};

const unsigned char Cara5 [] PROGMEM = {
 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x03, 0xff, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0x80, 0x00, 0x00, 0x03, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xf0, 0x00, 0x3f, 0xfc, 0x00, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0x80, 0x3f, 0xff, 0xfc, 0x01, 0xff, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xf8, 0x07, 0xff, 0xff, 0xff, 0xe0, 0x3f, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xe0, 0x7f, 0xff, 0xff, 0xff, 0xfc, 0x07, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0x03, 0xff, 0xff, 0xff, 0xff, 0xff, 0x81, 0xff, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xfc, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x7f, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xf0, 0x7f, 0x00, 0x00, 0x01, 0xfc, 0x1f, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xe1, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x03, 0xff, 0x07, 0xff, 0xff, 0xff,
 0xff, 0xff, 0x83, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xc1, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xfe, 0x0f, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xf0, 0xff, 0xff,
 0xff, 0xff, 0xfc, 0x3f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xf8, 0x7f, 0xff, 0xff,
 0xff, 0xff, 0xf8, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x1f, 0xff, 0xff,
 0xff, 0xff, 0xf0, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0x0f, 0xff, 0xff,
 0xff, 0xff, 0xe1, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x87, 0xff,
 0xff, 0xff, 0xc3, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xc3, 0xff, 0xff,
 0xff, 0xff, 0x87, 0xc0, 0x1f, 0xf0, 0x00, 0x00, 0x00, 0x0f, 0x80, 0x03, 0xe1, 0xff, 0xff,
 0xff, 0xff, 0x0f, 0x80, 0x7f, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x0f, 0xff,
 0xff, 0xff, 0x1f, 0x00, 0xff, 0x00, 0x00, 0x00, 0x01, 0xf8, 0x01, 0xf8, 0x7f, 0xff, 0xff,
 0xff, 0xff, 0xfe, 0x3e, 0x01, 0xff, 0x80, 0x00, 0x00, 0x07, 0xff, 0xfe, 0x00, 0x0f, 0x8f, 0x7f, 0xff,
 0xff, 0xfc, 0x3e, 0x01, 0xff, 0xfc, 0x00, 0x00, 0x0f, 0xff, 0xfe, 0x00, 0x00, 0x7c, 0x7f, 0xff,
 0xff, 0x80, 0x7c, 0x03, 0xff, 0xe0, 0x00, 0x00, 0x1f, 0xff, 0x00, 0x00, 0x00, 0x00, 0x03, 0xff,
 0xfc, 0x00, 0xfc, 0x03, 0xff, 0x0f, 0x00, 0x00, 0x1f, 0xff, 0x00, 0x00, 0x3e, 0x00, 0x7f,
 0xff, 0x8f, 0x30, 0xfc, 0x03, 0xff, 0xf0, 0x00, 0x00, 0x3f, 0xff, 0x00, 0x3e, 0x10, 0x1f,
 0xe0, 0xf0, 0xf8, 0x03, 0xff, 0x0f, 0xf8, 0x00, 0x00, 0x3f, 0xff, 0x00, 0x00, 0x1f, 0x0f, 0x0f,
 0xc3, 0xf1, 0xf8, 0x03, 0xff, 0xf8, 0x00, 0x00, 0x7f, 0xff, 0x00, 0x00, 0x1f, 0x0f, 0x87,
 0x87, 0xf1, 0xf8, 0x03, 0xff, 0xf8, 0x00, 0x00, 0x7f, 0xff, 0x00, 0x00, 0x1f, 0x0f, 0xc3,
 0x8f, 0xe1, 0xf8, 0x03, 0xff, 0xf8, 0x00, 0x00, 0x7f, 0xff, 0x00, 0x00, 0x0f, 0x8f, 0xe1,
 0x0f, 0xe1, 0xf8, 0x01, 0xff, 0xfc, 0x00, 0x00, 0x7f, 0xff, 0x00, 0x00, 0x0f, 0x8f, 0xf1,
 0x0f, 0xe1, 0xf8, 0x01, 0xff, 0xf8, 0x00, 0x00, 0x7f, 0xff, 0x00, 0x00, 0x1f, 0x87, 0xf1,
 0x1f, 0xe1, 0xf8, 0x00, 0x0f, 0xf8, 0x00, 0x00, 0x7f, 0xff, 0x00, 0x00, 0x1f, 0x87, 0xf1,
 0x1f, 0xe1, 0xfc, 0x00, 0x7f, 0xff, 0xf8, 0x00, 0x00, 0x7f, 0xff, 0x00, 0x00, 0x1f, 0x87, 0xf1,
 0x0f, 0xe1, 0xfc, 0x00, 0x3f, 0xff, 0xf8, 0x00, 0x00, 0x7f, 0xff, 0x00, 0x00, 0x1f, 0x87, 0xf1,
 0x8f, 0xe1, 0xfc, 0x00, 0x1f, 0xff, 0xf0, 0x00, 0x00, 0x3f, 0xff, 0x0f, 0x00, 0x1f, 0x8f, 0xe1,
 0x87, 0xf1, 0xfe, 0x00, 0x0f, 0xff, 0xf0, 0x00, 0x00, 0x3f, 0xff, 0xc0, 0x00, 0x3f, 0x0f, 0xe3,
 0xc3, 0xf1, 0xfe, 0x00, 0x03, 0xff, 0xe0, 0x00, 0x00, 0x1f, 0xff, 0x80, 0x00, 0x3f, 0x0f, 0xc3,
 0xe0, 0xf1, 0xff, 0x00, 0x00, 0xff, 0xc0, 0x00, 0x00, 0x0f, 0xfe, 0x00, 0x00, 0x7f, 0x0f, 0x07,
 0xff, 0x38, 0xff, 0x00, 0x00, 0x1f, 0x00, 0x00, 0x00, 0x07, 0xf8, 0x00, 0x00, 0x7e, 0x1c, 0x1f,
 0xfc, 0x00, 0xff, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x00, 0x3f,
 0xff, 0xb8, 0x7f, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xfc, 0x01, 0xff,
 0xff, 0xfe, 0x3f, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xfc, 0x3f, 0xff,
 0xff, 0xfe, 0x1f, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xf8, 0x7f, 0xff,
 0xff, 0x1f, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xf0, 0xff, 0xff,
}
```



```

const unsigned char PROGMEM phone4 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0xFF, 0x40, 0x00, 0x00,
0x00, 0x0F, 0xFF, 0xFF, 0xE8, 0x00, 0x00, 0x1F, 0xFE, 0x3F, 0xFE, 0x00, 0x00, 0x1C, 0x1F, 0x1B, 0xFF, 0x00, 0x00, 0x1E, 0x0F,
0xFF, 0x7F, 0x00, 0x00, 0x1C, 0x00, 0x5E, 0x07, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x07, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x0F, 0x00,
0x00, 0x1C, 0x00, 0x07, 0x00, 0x1C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x1C, 0x00,
0x00, 0x0E, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0E, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0E, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x0E, 0x00, 0x38, 0x00, 0x00, 0x0E, 0x00, 0x00, 0x38, 0x00, 0x00, 0x0E, 0x00, 0x00, 0x38, 0x00,
0x00, 0x1E, 0x00, 0x00, 0x38, 0x00, 0x14, 0x0E, 0x00, 0x00, 0x38, 0x00, 0xA, 0x1E, 0x00, 0x00, 0x38, 0x00, 0xA, 0x1E, 0x00,
0x00, 0x78, 0x00, 0x05, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x05, 0x1E, 0x00, 0x00, 0x78, 0x00, 0xA, 0x1C, 0x00, 0x00, 0x78, 0x00,
0xA, 0x1C, 0x00, 0x00, 0x70, 0x00, 0x14, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x70, 0x00, 0x00, 0x1C, 0x00,
0x00, 0x70, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x70, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x70, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x70, 0x00,
0x00, 0x3C, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x70, 0x00, 0x00, 0x38, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0xF0, 0x00, 0x00, 0x38, 0x00, 0x00, 0xE0, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x38, 0x00, 0x00, 0xE0, 0x00,
0x00, 0x38, 0x00, 0x00, 0xE0, 0x00, 0x00, 0x38, 0x00, 0x00, 0xFE, 0x80, 0x00, 0x38, 0x00, 0x00, 0xFF, 0xFF, 0xE8, 0x78, 0x00,
0x00, 0x7F, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x17, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x01, 0x7F, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00};
```

```
const unsigned char PROGMEM phone5 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x55, 0x55, 0x40, 0x00,
0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x3F, 0xF8, 0x77, 0xFC, 0x00, 0x00, 0x3F, 0xF8, 0x6F, 0xFC, 0x00, 0x00, 0x38, 0x3F,
0xFC, 0x1C, 0x00, 0x00, 0x78, 0xA0, 0xA0, 0x1E, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00,
```

```

const unsigned char PROGMEM phone7 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x55, 0x55, 0x40, 0x00,
0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x3F, 0xF8, 0x77, 0xFC, 0x00, 0x00, 0x3F, 0xF8, 0x6F, 0xFC, 0x00, 0x00, 0x38, 0x3F,
0xFC, 0x1E, 0x00, 0x00, 0x78, 0xA0, 0xA0, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00,
0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x78, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00,
0x00, 0x78, 0x00, 0x01, 0x00, 0x00, 0x00, 0x38, 0x00, 0x03, 0xC0, 0x00, 0x00, 0x78, 0x00, 0x03, 0xC0, 0x00, 0x00, 0x38, 0x00,
0x01, 0xE0, 0x00, 0x00, 0x78, 0x00, 0xE0, 0x00, 0x00, 0x38, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x78, 0x00, 0x00, 0x70, 0x00,
0x00, 0x38, 0x00, 0x00, 0x70, 0x00, 0x00, 0x78, 0x00, 0x00, 0x70, 0x00, 0x00, 0x38, 0x00, 0x00, 0x70, 0x00, 0x00, 0x78, 0x00,
0x00, 0xF0, 0x00, 0x00, 0x38, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x78, 0x00, 0x01, 0xE0, 0x00, 0x00, 0x38, 0x00, 0x03, 0xC0, 0x00,
0x00, 0x78, 0x00, 0x03, 0xC0, 0x00, 0x00, 0x38, 0x00, 0x01, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x38, 0x00,
0x00, 0x1E, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00,
0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x78, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00,
0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x1F, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x02, 0xAA, 0xAA, 0xA0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
```

```

const unsigned char PROGMEM phone8 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0xFF, 0x40, 0x00, 0x00,
0x00, 0x0F, 0xFF, 0xFF, 0xE8, 0x00, 0x00, 0x1F, 0xFE, 0x3F, 0xFE, 0x00, 0x00, 0x1C, 0x1F, 0x1B, 0xFF, 0x00, 0x00, 0x1E, 0x0F,
0xFF, 0x7F, 0x00, 0x00, 0x1C, 0x00, 0x5E, 0x07, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x07, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x0F, 0x00,
0x00, 0x1C, 0x00, 0x07, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x1C, 0x00,
0x00, 0x0F, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3C, 0x00, 0x00, 0xE0, 0x00,
0x00, 0x3C, 0x00, 0x01, 0xF0, 0x00, 0x00, 0x38, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x38, 0x00, 0x00, 0x78, 0x00, 0x00, 0x38, 0x00,
```

```
0x00, 0x38, 0x00, 0x00, 0x38, 0x00, 0x14, 0x3C, 0x00, 0x00, 0x38, 0x00, 0x0A, 0x3C, 0x00, 0x00, 0x38, 0x00, 0x0A, 0x1C, 0x00,
0x00, 0x78, 0x00, 0x05, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x05, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x0A, 0x1C, 0x00, 0x00, 0x78, 0x00,
0x0A, 0x3C, 0x00, 0x00, 0x70, 0x00, 0x14, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x38, 0x00, 0x00, 0x70, 0x00, 0x00, 0x78, 0x00,
0x00, 0x70, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x70, 0x00, 0x01, 0xF0, 0x00, 0x00, 0x70, 0x00, 0x00, 0xE0, 0x00, 0x00, 0x70, 0x00,
0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x70, 0x00, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0xF0, 0x00, 0x00, 0x38, 0x00, 0x00, 0xE0, 0x00, 0x00, 0x38, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x38, 0x00, 0x00, 0xE0, 0x00,
0x00, 0x38, 0x00, 0x00, 0xE0, 0x00, 0x00, 0x38, 0x00, 0x00, 0xFE, 0x80, 0x00, 0x38, 0x00, 0x00, 0xFF, 0xFF, 0xE8, 0x78, 0x00,
0x00, 0x7F, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x17, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0x01, 0x7F, 0xF0, 0x00, 0x00, 0x00, 0x00};
```

```

const unsigned char PROGMEM phone9 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x55, 0x55, 0x40, 0x00,
0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x3F, 0xF8, 0x77, 0xFC, 0x00, 0x00, 0x3F, 0xF8, 0x6F, 0xFC, 0x00, 0x00, 0x38, 0x3F,
0xFC, 0x1E, 0x00, 0x00, 0x78, 0xA0, 0xA0, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00,
0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00,
0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x20, 0x00, 0x00, 0x78, 0x00, 0x00, 0x78, 0x00, 0x00, 0x38, 0x00, 0x00, 0x7C, 0x00,
0x00, 0x78, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x78, 0x00, 0x00, 0x0E, 0x00, 0x00, 0x38, 0x00,
0x08, 0x0F, 0x00, 0x00, 0x78, 0x00, 0x1E, 0x0F, 0x00, 0x00, 0x38, 0x00, 0x1E, 0x07, 0x00, 0x00, 0x78, 0x00, 0x0F, 0x07, 0x00,
0x00, 0x38, 0x00, 0x07, 0x07, 0x00, 0x00, 0x78, 0x00, 0x07, 0x07, 0x00, 0x00, 0x38, 0x00, 0x0F, 0x07, 0x00, 0x00, 0x78, 0x00,
0x1E, 0x07, 0x00, 0x00, 0x38, 0x00, 0x1E, 0x0F, 0x00, 0x00, 0x78, 0x00, 0x08, 0x0F, 0x00, 0x00, 0x38, 0x00, 0x00, 0x0E, 0x00,
0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x78, 0x00, 0x00, 0x38, 0x00,
0x00, 0x78, 0x00, 0x00, 0x78, 0x00, 0x20, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00,
0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x00, 0x00, 0x3F, 0xFF, 0xFC, 0x00,
0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x1F, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x02, 0xAA, 0xAA, 0xA0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

```

```

const unsigned char PROGMEM phone10 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0xFF, 0xF0, 0x00,
0x00, 0x17, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x7F, 0xF4, 0xDF, 0xF8, 0x00, 0x00, 0xFF, 0xE1, 0xF8, 0x38, 0x00, 0x00, 0xFD, 0xFF,
0xF0, 0x78, 0x00, 0x00, 0xE0, 0x3D, 0x00, 0x38, 0x00, 0x00, 0xE0, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00, 0x00,
0x00, 0xE0, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x70, 0x00,
0x00, 0x1C, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x70, 0x00, 0x00, 0x1F, 0x00, 0x00, 0x70, 0x00, 0x00, 0x0F, 0x00,
0x00, 0x70, 0x00, 0x00, 0x07, 0x80, 0x00, 0x70, 0x00, 0x00, 0x07, 0x80, 0x00, 0x70, 0x00, 0x06, 0x03, 0x80, 0x00, 0x78, 0x00,
0x0F, 0x83, 0xC0, 0x00, 0x70, 0x00, 0x07, 0x83, 0xC0, 0x00, 0x78, 0x00, 0x03, 0xC1, 0xC0, 0x00, 0x78, 0x00, 0x03, 0xC1,
0x00, 0x38, 0x00, 0x01, 0xC1, 0xC0, 0x00, 0x78, 0x00, 0x01, 0xC1, 0xC0, 0x00, 0x38, 0x00, 0x03, 0xC1, 0xC0, 0x00, 0x38, 0x00,
0x03, 0xC1, 0xC0, 0x00, 0x38, 0x00, 0x07, 0x81, 0xC0, 0x00, 0x38, 0x00, 0x0F, 0x83, 0xC0, 0x00, 0x38, 0x00, 0x06, 0x03, 0x80,
0x00, 0x38, 0x00, 0x00, 0x07, 0x80, 0x00, 0x3C, 0x00, 0x00, 0x07, 0x80, 0x00, 0x38, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x1E, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x3E, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x1C, 0x00,
0x00, 0x07, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00, 0x1C, 0x00, 0x01, 0x7F, 0x00, 0x00, 0x1E, 0x17, 0xFF, 0xFF,
0x00, 0x1F, 0xFF, 0xFF, 0xFE, 0x00, 0x00, 0x0F, 0xFF, 0xFF, 0xE8, 0x00, 0x00, 0x0F, 0xFF, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00};
```

```

const unsigned char PROGMEM phone11 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x55, 0x55, 0x40, 0x00,
0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x3F, 0xF8, 0x77, 0xFC, 0x00, 0x00, 0x3F, 0xF8, 0x6F, 0xFC, 0x00, 0x00, 0x38, 0x3F,
0xFC, 0x1E, 0x00, 0x00, 0x78, 0xA0, 0xA0, 0x1C, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00,
0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x06, 0x00, 0x00, 0x38, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x78, 0x00,
0x00, 0x0F, 0x80, 0x00, 0x38, 0x00, 0x00, 0x07, 0x80, 0x00, 0x78, 0x00, 0x00, 0x03, 0xC0, 0x00, 0x38, 0x00, 0x00, 0x01, 0xE0,
0x00, 0x78, 0x00, 0x01, 0x01, 0xE0, 0x00, 0x38, 0x00, 0x03, 0xC0, 0xE0, 0x00, 0x78, 0x00, 0x03, 0xC0, 0xE0, 0x00, 0x38, 0x00,
0x01, 0xE0, 0xF0, 0x00, 0x78, 0x00, 0xE0, 0x70, 0x00, 0x38, 0x00, 0x00, 0xF0, 0x70, 0x00, 0x78, 0x00, 0x00, 0x70, 0x70,
0x00, 0x38, 0x00, 0x00, 0x70, 0x70, 0x00, 0x78, 0x00, 0x00, 0x70, 0x70, 0x00, 0x38, 0x00, 0x00, 0x70, 0x70, 0x00, 0x78, 0x00,
0x00, 0xF0, 0x70, 0x00, 0x38, 0x00, 0xF0, 0x70, 0x00, 0x78, 0x00, 0x01, 0xE0, 0xF0, 0x00, 0x38, 0x00, 0x03, 0xC0, 0xF0,
```

```
0x00, 0x78, 0x00, 0x03, 0xC0, 0xE0, 0x00, 0x38, 0x00, 0x01, 0x01, 0xE0, 0x00, 0x78, 0x00, 0x00, 0x01, 0xE0, 0x00, 0x38, 0x00,
0x00, 0x03, 0xC0, 0x00, 0x78, 0x00, 0x00, 0x07, 0x80, 0x00, 0x38, 0x00, 0x00, 0xF0, 0x80, 0x00, 0x78, 0x00, 0x00, 0x0F, 0x00,
0x00, 0x38, 0x00, 0x00, 0x04, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00,
0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x1F, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x02, 0xAA, 0xAA, 0xA0, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00};
```

```

const unsigned char PROGMEM phone12 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0xFF, 0x40, 0x00, 0x00,
0x00, 0x0F, 0xFF, 0xE8, 0x00, 0x00, 0x1F, 0xFE, 0x3F, 0xFE, 0x00, 0x00, 0x1C, 0x1F, 0x1B, 0xFF, 0x00, 0x00, 0x1E, 0x0F,
0xFF, 0xFF, 0x00, 0x00, 0x1C, 0x00, 0xBE, 0x07, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x07, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x0F, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x04, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0A, 0x00, 0x00, 0x1C, 0x00,
0x00, 0x05, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x02, 0x80, 0x00, 0x3C, 0x00, 0x00, 0x01, 0x40, 0x00, 0x3C, 0x00, 0x00, 0xE1, 0x80,
0x00, 0x38, 0x00, 0x01, 0xF0, 0x40, 0x00, 0x3C, 0x00, 0x00, 0xF0, 0xA0, 0x00, 0x38, 0x00, 0x00, 0x78, 0xA0, 0x00, 0x38, 0x00,
0x00, 0x38, 0x50, 0x00, 0x38, 0x00, 0x00, 0x3C, 0x20, 0x00, 0x38, 0x00, 0x00, 0x3C, 0x50, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x50,
0x00, 0x38, 0x00, 0x00, 0x1C, 0x20, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x50, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x50, 0x00, 0x78, 0x00,
0x00, 0x3C, 0x20, 0x00, 0x70, 0x00, 0x00, 0x3C, 0x50, 0x00, 0x78, 0x00, 0x00, 0x38, 0x60, 0x00, 0x00, 0x70, 0x00, 0x00, 0x78, 0xA0,
0x00, 0x70, 0x00, 0x00, 0xF0, 0x40, 0x00, 0x70, 0x00, 0x01, 0xF0, 0xA0, 0x00, 0x70, 0x00, 0x00, 0xE1, 0x40, 0x00, 0x70, 0x00,
0x00, 0x01, 0x80, 0x00, 0xF0, 0x00, 0x00, 0x05, 0x00, 0x00, 0x70, 0x00, 0x00, 0x0A, 0x80, 0x00, 0xF0, 0x00, 0x00, 0x06, 0x00,
0x00, 0xF0, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x00,
0x00, 0x00, 0x00, 0xE0, 0x00, 0x00, 0x38, 0x00, 0x00, 0xFE, 0x80, 0x00, 0x38, 0x00, 0x00, 0xFF, 0xFF, 0xE8, 0x78, 0x00,
0x00, 0x7F, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x17, 0xFF, 0xF0, 0x00, 0x00, 0x01, 0x7F, 0xF0, 0x00, 0x00, 0x00, 0x00};
```

```

const unsigned char PROGMEM phone13 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x55, 0x55, 0x50, 0x00,
0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x3F, 0xF8, 0x77, 0xFC, 0x00, 0x00, 0x7F, 0xF8, 0x6F, 0xFC, 0x00, 0x00, 0x38, 0x3F,
0xFC, 0x1C, 0x00, 0x00, 0x78, 0xA0, 0xA0, 0x1E, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00,
0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00,
0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x20, 0x00, 0x00, 0x78, 0x00, 0x00, 0x78, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x7C, 0x00,
0x00, 0x78, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x78, 0x00, 0x00, 0x0E, 0x00, 0x00, 0x38, 0x00,
0x00, 0x0F, 0x00, 0x00, 0x78, 0x00, 0x14, 0x0F, 0x00, 0x00, 0x38, 0x00, 0xA, 0x07, 0x00, 0x00, 0x78, 0x00, 0xA, 0x07, 0x00,
0x00, 0x38, 0x00, 0x05, 0x07, 0x00, 0x00, 0x78, 0x00, 0x05, 0x07, 0x00, 0x00, 0x38, 0x00, 0xA, 0x07, 0x00, 0x00, 0x78, 0x00,
0xA, 0x07, 0x00, 0x00, 0x38, 0x00, 0x14, 0x0F, 0x00, 0x00, 0x78, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x38, 0x00, 0x00, 0x0E, 0x00,
0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x38, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x78, 0x00, 0x00, 0x38, 0x00,
0x00, 0x78, 0x00, 0x00, 0x78, 0x00, 0x20, 0x00, 0x00, 0x38, 0x00, 0x00,
0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x00, 0x00, 0x3F, 0xFF, 0xFF, 0xFC,
0x00, 0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x02, 0xAA, 0xAA, 0xA0, 0x00, 0x00, 0x00, 0x00, 0x00};
```

```

const unsigned char PROGMEM phone14 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0xFF, 0xF0, 0x00,
0x00, 0x17, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x7F, 0xF4, 0xDF, 0xF8, 0x00, 0x00, 0xFF, 0xE1, 0xF8, 0x38, 0x00, 0x00, 0xFD, 0xFF,
0xF0, 0x78, 0x00, 0x00, 0xE0, 0x3D, 0x00, 0x38, 0x00, 0x00, 0xE0, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00, 0x00,
0x00, 0xE0, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x70, 0x00,
0x00, 0x1C, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x70, 0x00, 0x00, 0x1F, 0x00, 0x00, 0x70, 0x00, 0x00, 0x0F, 0x00,
0x00, 0x70, 0x00, 0x00, 0x07, 0x80, 0x00, 0x70, 0x00, 0x00, 0x07, 0x80, 0x00, 0x70, 0x00, 0x00, 0x03, 0x80, 0x00, 0x78, 0x00,
0x08, 0x03, 0xC0, 0x00, 0x70, 0x00, 0x1E, 0x03, 0xC0, 0x00, 0x78, 0x00, 0x1E, 0x01, 0xC0, 0x00, 0x78, 0x00, 0x0F, 0x01, 0xC0,
0x00, 0x38, 0x00, 0x0F, 0x01, 0xC0, 0x00, 0x78, 0x00, 0x07, 0x01, 0xC0, 0x00, 0x38, 0x00, 0x0F, 0x01, 0xC0, 0x00, 0x38, 0x00,
0x1E, 0x01, 0xC0, 0x00, 0x38, 0x00, 0x1E, 0x01, 0xC0, 0x00, 0x38, 0x00, 0x08, 0x03, 0xC0, 0x00, 0x38, 0x00, 0x03, 0x80,
0x00, 0x38, 0x00, 0x00, 0x07, 0x80, 0x00, 0x3C, 0x00, 0x00, 0x07, 0x80, 0x00, 0x38, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x1E, 0x00, 0x00, 0x3C, 0x00, 0x3E, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x1C, 0x00,
```

```
0x00, 0x07, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x07, 0x00, 0x00, 0x1C, 0x00, 0x01, 0x7F, 0x00, 0x00, 0x1E, 0x17, 0xFF, 0xFF, 0x00,
0x00, 0x1F, 0xFF, 0xFF, 0xFE, 0x00, 0x00, 0x0F, 0xFF, 0xFF, 0xE8, 0x00, 0x00, 0x0F, 0xFF, 0x40, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00};
```

```

const unsigned char PROGMEM phone15 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x55, 0x55, 0x40, 0x00,
0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x3F, 0xF8, 0x77, 0xFC, 0x00, 0x00, 0x3F, 0xF8, 0x6F, 0xFC, 0x00, 0x00, 0x38, 0x3F,
0xFC, 0x1E, 0x00, 0x00, 0x78, 0xA0, 0xA0, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00,
0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x06, 0x00, 0x00, 0x38, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x78, 0x00,
0x00, 0x0F, 0x80, 0x00, 0x38, 0x00, 0x00, 0x07, 0x80, 0x00, 0x78, 0x00, 0x00, 0x03, 0xC0, 0x00, 0x38, 0x00, 0x00, 0x01, 0xE0,
0x00, 0x78, 0x00, 0x00, 0x01, 0xE0, 0x00, 0x38, 0x00, 0x00, 0x00, 0xE0, 0x00, 0x78, 0x00, 0x06, 0x00, 0xE0, 0x00, 0x38, 0x00,
0x0F, 0x80, 0xF0, 0x00, 0x78, 0x00, 0x07, 0x80, 0x70, 0x00, 0x38, 0x00, 0x03, 0xC0, 0x70, 0x00, 0x78, 0x00, 0x03, 0xC0, 0x70,
0x00, 0x38, 0x00, 0x01, 0xC0, 0x70, 0x00, 0x78, 0x00, 0x01, 0xC0, 0x70, 0x00, 0x38, 0x00, 0x03, 0xC0, 0x70, 0x00, 0x78, 0x00,
0x03, 0xC0, 0x70, 0x00, 0x38, 0x00, 0x07, 0x80, 0x70, 0x00, 0x78, 0x00, 0x0F, 0x00, 0xF0, 0x00, 0x38, 0x00, 0x06, 0x00, 0xF0,
0x00, 0x78, 0x00, 0x00, 0xE0, 0x00, 0x38, 0x00, 0x00, 0x01, 0xE0, 0x00, 0x78, 0x00, 0x00, 0x01, 0xE0, 0x00, 0x38, 0x00,
0x00, 0x03, 0xC0, 0x00, 0x78, 0x00, 0x07, 0x80, 0x00, 0x38, 0x00, 0x00, 0x0F, 0x80, 0x00, 0x78, 0x00, 0x00, 0x0F, 0x00,
0x00, 0x38, 0x00, 0x00, 0x04, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00,
0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x1F, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x02, 0xAA, 0xAA, 0xA0, 0x00, 0x00, 0x00, 0x00, 0x00};
```

```

const unsigned char PROGMEM phone16 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0xFF, 0x40, 0x00, 0x00,
0x00, 0x0F, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x1F, 0xFE, 0x3F, 0xFE, 0x00, 0x00, 0x1C, 0x1F, 0x1B, 0xFF, 0x00, 0x00, 0x1E, 0x0F,
0xFF, 0x00, 0x00, 0x1C, 0x00, 0xBE, 0x07, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x07, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x0F, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x04, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0A, 0x00, 0x00, 0x1C, 0x00,
0x00, 0x05, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x02, 0x80, 0x00, 0x3C, 0x00, 0x00, 0x01, 0x40, 0x00, 0x3C, 0x00, 0x00, 0x01, 0x80,
0x00, 0x38, 0x00, 0x01, 0x00, 0x40, 0x00, 0x3C, 0x00, 0x03, 0xC0, 0xA0, 0x00, 0x38, 0x00, 0x03, 0xE0, 0xA0, 0x00, 0x38, 0x00,
0x01, 0xE0, 0x50, 0x00, 0x38, 0x00, 0xF0, 0x20, 0x00, 0x38, 0x00, 0x00, 0xF0, 0x50, 0x00, 0x78, 0x00, 0x00, 0xF0, 0x50,
0x00, 0x38, 0x00, 0x00, 0x70, 0x20, 0x00, 0x78, 0x00, 0x00, 0x70, 0x50, 0x00, 0x78, 0x00, 0x00, 0x70, 0x50, 0x00, 0x78, 0x00,
0x00, 0xF0, 0x20, 0x00, 0x78, 0x00, 0x00, 0xF0, 0x50, 0x00, 0x78, 0x00, 0x01, 0xE0, 0x30, 0x00, 0x70, 0x00, 0x03, 0xC0, 0x40,
0x00, 0x70, 0x00, 0x03, 0xC0, 0xA0, 0x00, 0x70, 0x00, 0x01, 0x00, 0xA0, 0x00, 0x70, 0x00, 0x00, 0x01, 0x40, 0x00, 0x70, 0x00,
0x00, 0x01, 0x40, 0x00, 0xF0, 0x00, 0x00, 0x02, 0x80, 0x00, 0x70, 0x00, 0x00, 0x0B, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x06, 0x00,
0x00, 0xF0, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x00,
0x00, 0x00, 0x00, 0x00, 0xE0, 0x00, 0x00, 0x38, 0x00, 0x00, 0xFE, 0x80, 0x00, 0x38, 0x00, 0x00, 0xFF, 0xFF, 0xE8, 0x78, 0x00,
0x00, 0x7F, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x17, 0xFF, 0xFF, 0xF0, 0x00, 0x00, 0x02, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
```

```

const unsigned char PROGMEM phone17 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x55, 0x55, 0x40, 0x00,
0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x3F, 0xF8, 0x77, 0xFC, 0x00, 0x00, 0x3F, 0xF8, 0x6F, 0xFC, 0x00, 0x00, 0x38, 0x3F,
0xFC, 0x1E, 0x00, 0x00, 0x78, 0xA0, 0xA0, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00,
0x00, 0x38, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0xE0, 0x00,
0x00, 0x78, 0x00, 0x01, 0xF0, 0x00, 0x00, 0x38, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x78, 0x00, 0x00, 0x78, 0x00, 0x00, 0x38, 0x00,
0x00, 0x38, 0x00, 0x78, 0x00, 0x3C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00,
0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x38, 0x00, 0x00, 0x38, 0x00, 0x00, 0x78, 0x00,
0x00, 0x78, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x38, 0x00, 0x01, 0xF0, 0x00, 0x00, 0x78, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x38, 0x00,
0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00,
0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x1F, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x02, 0xAA, 0xAA, 0xA0, 0x00, 0x00, 0x00, 0x00, 0x00};
```

```

const unsigned char PROGMEM phone18 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x55, 0x55, 0x40, 0x00,
0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x3F, 0xF8, 0x77, 0xFC, 0x00, 0x00, 0x3F, 0xF8, 0x6F, 0xFC, 0x00, 0x00, 0x38, 0x3F,
0xFC, 0x1E, 0x00, 0x00, 0x78, 0xA0, 0xA0, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00,
0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00,
0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x20, 0x00, 0x00, 0x78, 0x00, 0x00, 0x78, 0x00, 0x00, 0x38, 0x00, 0x00, 0x7C, 0x00,
0x00, 0x78, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x78, 0x00, 0x00, 0x0E, 0x00, 0x00, 0x38, 0x00,
0x00, 0x0F, 0x00, 0x00, 0x78, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x38, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x07, 0x00,
0x00, 0x38, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x07, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00,
0x00, 0x07, 0x00, 0x00, 0x38, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x78, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x38, 0x00, 0x00, 0x0E, 0x00,
0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x38, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x78, 0x00, 0x00, 0x38, 0x00,
0x00, 0x78, 0x00, 0x00, 0x78, 0x00, 0x20, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00,
0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x00, 0x00, 0x3F, 0xFF, 0xFC, 0x00,
0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x1F, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x02, 0xAA, 0xAA, 0xA0, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00};
```

```

const unsigned char PROGMEM phone20 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x55, 0x55, 0x50, 0x00,
0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x3F, 0xF8, 0x77, 0xFC, 0x00, 0x00, 0x7F, 0xF8, 0x6F, 0xFC, 0x00, 0x00, 0x38, 0x3F,
0xFC, 0x1C, 0x00, 0x00, 0x78, 0xA0, 0xA0, 0x1E, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00,
0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x06, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x78, 0x00,
0x00, 0x0F, 0x80, 0x00, 0x38, 0x00, 0x00, 0x07, 0x80, 0x00, 0x78, 0x00, 0x00, 0x03, 0xC0, 0x00, 0x38, 0x00, 0x00, 0x01, 0xE0,
0x00, 0x78, 0x00, 0x00, 0x01, 0xE0, 0x00, 0x38, 0x00, 0x00, 0x00, 0xE0, 0x00, 0x78, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x38, 0x00,
0x00, 0xF0, 0x00, 0x78, 0x00, 0x00, 0x70, 0x00, 0x38, 0x00, 0x00, 0x00, 0x70, 0x00, 0x78, 0x00, 0x00, 0x00, 0x70,
0x00, 0x38, 0x00, 0x00, 0x00, 0x70, 0x00, 0x78, 0x00, 0x00, 0x00, 0x70, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x70, 0x00, 0x78, 0x00,
0x00, 0x00, 0x70, 0x00, 0x38, 0x00, 0x00, 0x00, 0x70, 0x00, 0x78, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0xF0,
0x00, 0x78, 0x00, 0x00, 0x00, 0xE0, 0x00, 0x38, 0x00, 0x00, 0x01, 0xE0, 0x00, 0x78, 0x00, 0x00, 0x01, 0xE0, 0x00, 0x38, 0x00,
0x00, 0x03, 0xC0, 0x00, 0x78, 0x00, 0x00, 0x07, 0x80, 0x00, 0x38, 0x00, 0x00, 0x0F, 0x80, 0x00, 0x78, 0x00, 0x00, 0x0F, 0x00,
0x00, 0x38, 0x00, 0x00, 0x04, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x01, 0xE0, 0x00, 0x00, 0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00,
0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x01F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x02, 0xAA, 0xAA, 0xA0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
```

```
const unsigned char PROGMEM phone21 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x55, 0x55, 0x50, 0x00, 0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x3F, 0xF8, 0x6F, 0xFC, 0x00, 0x00, 0x7F, 0xF8, 0x7F, 0xFC, 0x00, 0x00, 0x38, 0x3F,
```

```
const unsigned char PROGMEM phone24 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x55, 0x55, 0x40, 0x00,
0x00, 0x3F, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x3F, 0xF8, 0x77, 0xFC, 0x00, 0x00, 0x3F, 0xF8, 0x6F, 0xFE, 0x00, 0x00, 0x38, 0x3F,
0xFC, 0x1C, 0x00, 0x00, 0x78, 0x0A, 0xA0, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00,
0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x1E, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00,
```

```
0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x38, 0x00,
0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00,
0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00,
0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x08, 0x00,
0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x38, 0x00,
0x00, 0x1E, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00,
0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x78, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00,
0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x1F, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x02, 0xAA, 0xAA, 0xA0, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00};
```

```

const unsigned char PROGMEM phone27 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x55, 0x55, 0x40, 0x00,
0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x3F, 0xF8, 0x77, 0xFC, 0x00, 0x00, 0x3F, 0xF8, 0x6F, 0xFC, 0x00, 0x00, 0x38, 0x3F,
0xFC, 0x1C, 0x00, 0x00, 0x78, 0xA0, 0xA0, 0x1E, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00,
0x00, 0x38, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00,
0x00, 0x1E, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1E, 0x00,
0x00, 0x78, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x38, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00,
0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00,
```

```
0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00,
0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x38, 0x00,
0x00, 0x1E, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00,
0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x78, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00,
0x00, 0x3F, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x1F, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x02, 0xAA, 0xA0, 0x00, 0x00, 0x00, 0x00, 0x00};
```

//////////////////reloj

```
const unsigned char PROGMEM frame0 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x3C, 0x00,
0x01, 0xFF, 0x00, 0x00, 0xFF, 0x80, 0x07, 0xEF, 0x00, 0x00, 0xFB, 0xE0, 0x0F, 0x0E, 0x00, 0x00, 0x70, 0xF0, 0x1E, 0x3C, 0x1F,
0xF0, 0x3C, 0x30, 0x18, 0x38, 0x7F, 0xFF, 0x1C, 0x38, 0x18, 0xF3, 0xF8, 0x2F, 0xCF, 0x1C, 0x38, 0xE7, 0xC0, 0x03, 0xE7, 0x1C,
0x33, 0xDE, 0x00, 0x00, 0x7B, 0xCC, 0x33, 0xBC, 0x00, 0x00, 0x3D, 0xCC, 0x3F, 0x78, 0x00, 0x00, 0x1E, 0xFC, 0x3F, 0x70,
0x00, 0x00, 0x0E, 0x7C, 0x3C, 0xE0, 0x00, 0x00, 0x07, 0x3C, 0x39, 0xC0, 0x00, 0x00, 0x03, 0x9C, 0x11, 0x82, 0x00, 0x00, 0x01,
0x88, 0x03, 0x87, 0x00, 0x01, 0xC0, 0x03, 0x03, 0x80, 0x00, 0x00, 0xC0, 0x07, 0x01, 0xC0, 0x00, 0x00, 0xE0, 0x06, 0x00,
0xE0, 0x00, 0x00, 0x60, 0x0E, 0x00, 0x70, 0x00, 0x00, 0x60, 0x0E, 0x00, 0x38, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x1C, 0x00, 0x00,
0x30, 0x0C, 0x00, 0x0E, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x7F, 0x80, 0x00, 0x30, 0x0C, 0x01, 0xFF, 0x80, 0x00, 0x30, 0x0C, 0x00,
0x55, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00,
0x30, 0x0E, 0x00, 0x00, 0x00, 0x70, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
```

```
const unsigned char PROGMEM frame1 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x3C, 0x00,
0x01, 0xFF, 0x00, 0x00, 0xFF, 0x80, 0x07, 0xEF, 0x00, 0x00, 0xFB, 0xE0, 0x0F, 0x0E, 0x00, 0x00, 0x70, 0xF0, 0x1E, 0x3C, 0x1F,
0xF0, 0x3C, 0x30, 0x18, 0x38, 0x7F, 0xFF, 0x1C, 0x38, 0x18, 0xF3, 0xF8, 0x2F, 0xCF, 0x1C, 0x38, 0xE7, 0xC0, 0x03, 0xE7, 0x1C,
0x33, 0xDE, 0x00, 0x00, 0x7B, 0xCC, 0x33, 0xBC, 0x00, 0x00, 0x3D, 0xCC, 0x3F, 0x78, 0x00, 0x00, 0x1E, 0xFC, 0x3F, 0x70,
0x00, 0x00, 0x0E, 0x7C, 0x3C, 0xE0, 0x00, 0x00, 0x07, 0x3C, 0x39, 0xC0, 0x00, 0x00, 0x03, 0x9C, 0x11, 0x82, 0x00, 0x00, 0x01,
0x88, 0x03, 0x81, 0xC0, 0x00, 0x01, 0xC0, 0x03, 0x00, 0xE0, 0x00, 0x00, 0xC0, 0x07, 0x00, 0xE0, 0x00, 0x00, 0xE0, 0x06, 0x00,
0x70, 0x00, 0x00, 0x60, 0x0E, 0x00, 0x38, 0x00, 0x00, 0x60, 0x0E, 0x00, 0x1C, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x0E, 0x00, 0x00,
0x30, 0x0C, 0x00, 0x07, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x7F, 0x80, 0x00, 0x30, 0x0C, 0x01, 0xFF, 0x80, 0x01, 0xF3, 0x80, 0x03,
0x83, 0xF0, 0x1F, 0xC1, 0xC0, 0x07, 0x00, 0xFF, 0x00, 0xE0, 0x06, 0x00, 0x1F, 0xF8, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
```

```
const unsigned char PROGMEM frame2 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x3C, 0x00,
0x01, 0xFF, 0x00, 0x00, 0xFF, 0x80, 0x07, 0xEF, 0x00, 0x00, 0xFB, 0xE0, 0x0F, 0x0E, 0x00, 0x00, 0x70, 0xF0, 0x1E, 0x3C, 0x1F,
0xF0, 0x3C, 0x38, 0x18, 0x38, 0x7F, 0xFF, 0x1C, 0x38, 0x18, 0xF3, 0xF8, 0x2F, 0xCF, 0x1C, 0x38, 0xE7, 0xC0, 0x03, 0xE7, 0x1C,
0x33, 0xDE, 0x00, 0x00, 0x7B, 0xCC, 0x33, 0xBC, 0x00, 0x00, 0x3D, 0xCC, 0x3F, 0x78, 0x00, 0x00, 0x1E, 0xFC, 0x3F, 0x70,
0x00, 0x00, 0x0E, 0x7C, 0x3C, 0xE0, 0x00, 0x00, 0x07, 0x3C, 0x39, 0xC0, 0x00, 0x00, 0x03, 0x9C, 0x11, 0x80, 0xE0, 0x00, 0x01,
0x88, 0x03, 0x80, 0xE0, 0x00, 0x01, 0xC0, 0x03, 0x00, 0x70, 0x00, 0x00, 0xE0, 0x07, 0x00, 0x30, 0x00, 0x00, 0xE0, 0x06, 0x00,
0x38, 0x00, 0x00, 0x60, 0x0E, 0x00, 0x1C, 0x00, 0x00, 0x60, 0x0E, 0x00, 0x0C, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x0E, 0x00, 0x00,
0x30, 0x0C, 0x00, 0x07, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x7F, 0x80, 0x00, 0x30, 0x0C, 0x01, 0xFF, 0x80, 0x00, 0x30, 0x0C, 0x00,
0x55, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00};
```

```

const unsigned char PROGMEM frame3 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x3C, 0x00,
0x01, 0xFF, 0x00, 0x00, 0xFF, 0x80, 0x07, 0xEF, 0x00, 0x00, 0xFB, 0xE0, 0x0F, 0x0E, 0x00, 0x00, 0x70, 0xF0, 0x1E, 0x3C, 0x1F,
0xF0, 0x3C, 0x30, 0x18, 0x38, 0x7F, 0xFF, 0x1C, 0x38, 0x18, 0xF3, 0xF8, 0x2F, 0xCF, 0x1C, 0x38, 0xE7, 0xC0, 0x03, 0xE7, 0x1C,
0x33, 0xDE, 0x00, 0x00, 0x7B, 0xCC, 0x33, 0xBC, 0x00, 0x00, 0x3D, 0xCC, 0x3F, 0x78, 0x00, 0x00, 0x00, 0x1E, 0xFC, 0x3F, 0x70,
0x00, 0x00, 0x0E, 0x7C, 0x3C, 0xE0, 0x60, 0x00, 0x07, 0x3C, 0x39, 0xC0, 0x60, 0x00, 0x03, 0x9C, 0x11, 0x80, 0x70, 0x00, 0x01,
0x88, 0x03, 0x80, 0x30, 0x00, 0x01, 0xC0, 0x03, 0x00, 0x38, 0x00, 0x00, 0xC0, 0x07, 0x00, 0x18, 0x00, 0x00, 0xE0, 0x06, 0x00,
0x1C, 0x00, 0x00, 0x60, 0x0E, 0x00, 0x0C, 0x00, 0x00, 0x60, 0x0E, 0x00, 0x0E, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x06, 0x00, 0x00,
0x30, 0x0C, 0x00, 0x03, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x7F, 0x80, 0x00, 0x30, 0x0C, 0x01, 0xFF, 0x80, 0x00, 0x30, 0x0C, 0x00,
0x55, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00,
0x30, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x06, 0x00, 0x00, 0x00, 0x00, 0x60, 0x06, 0x00, 0x00, 0x00, 0x00, 0x60, 0x07, 0x00,
0x00, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x03, 0x80, 0x00, 0x00, 0x01, 0xC0, 0x01, 0xC0, 0x00, 0x00, 0x03,
0x80, 0x01, 0xC0, 0x00, 0x03, 0x80, 0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x70, 0x00, 0x00, 0x0E, 0x00, 0x00, 0x78,
0x00, 0x00, 0x1E, 0x00, 0x00, 0xFE, 0x00, 0x00, 0x7F, 0x00, 0x01, 0xCF, 0x80, 0x01, 0xF3, 0x80, 0x03, 0x83, 0xF0, 0x1F, 0xC1,
0xC0, 0x07, 0x00, 0xFF, 0xFF, 0x00, 0xE0, 0x06, 0x00, 0x1F, 0xF8, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00};
```

```

const unsigned char PROGMEM frame4 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x3C, 0x00,
0x01, 0xFF, 0x00, 0x00, 0xFF, 0x80, 0x07, 0xEF, 0x00, 0x00, 0xFB, 0xE0, 0x0F, 0x0E, 0x00, 0x00, 0x70, 0xF0, 0x1E, 0x3C, 0x1F,
0xF0, 0x3C, 0x30, 0x18, 0x38, 0x7F, 0xFF, 0x1C, 0x38, 0x18, 0xF3, 0xF8, 0x2F, 0xCF, 0x1C, 0x38, 0xE7, 0xC0, 0x03, 0xE7, 0x1C,
0x33, 0xDE, 0x00, 0x00, 0x7B, 0xCC, 0x33, 0xBC, 0x00, 0x00, 0x3D, 0xCC, 0x3F, 0x78, 0x00, 0x00, 0x1E, 0xFC, 0x3F, 0x70,
0x10, 0x00, 0x0E, 0x7C, 0x3C, 0xE0, 0x30, 0x00, 0x07, 0x3C, 0x39, 0xC0, 0x38, 0x00, 0x03, 0x9C, 0x11, 0x80, 0x18, 0x00, 0x01,
0x88, 0x03, 0x80, 0x18, 0x00, 0x01, 0xC0, 0x03, 0x00, 0x1C, 0x00, 0x00, 0xC0, 0x07, 0x00, 0x0C, 0x00, 0x00, 0xE0, 0x06, 0x00,
0x0E, 0x00, 0x00, 0x60, 0x0E, 0x00, 0x06, 0x00, 0x00, 0x60, 0x0E, 0x00, 0x06, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x07, 0x00, 0x00,
0x30, 0x0C, 0x00, 0x03, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x7F, 0x80, 0x00, 0x30, 0x0C, 0x01, 0xFF, 0x80, 0x00, 0x30, 0x0C, 0x00,
0x55, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00,
0x30, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x06, 0x00, 0x00, 0x00, 0x00, 0x60, 0x06, 0x00, 0x00, 0x00, 0x00, 0x60, 0x07, 0x00,
0x00, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x03, 0x80, 0x00, 0x00, 0x01, 0xC0, 0x01, 0xC0, 0x00, 0x00, 0x03,
0x80, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x70, 0x00, 0x00, 0x0E, 0x00, 0x00, 0x78,
0x00, 0x00, 0x1E, 0x00, 0x00, 0xFE, 0x00, 0x00, 0x7F, 0x00, 0x01, 0xCF, 0x80, 0x01, 0xF3, 0x80, 0x03, 0x83, 0xF0, 0x1F, 0xC1,
0xC0, 0x07, 0x00, 0xFF, 0xFF, 0x00, 0xE0, 0x06, 0x00, 0x1F, 0xF8, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00};
```

```

const unsigned char PROGMEM frame5 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x3C, 0x00,
0x01, 0xFF, 0x00, 0x00, 0xFF, 0x80, 0x07, 0xEF, 0x00, 0x00, 0xFB, 0xE0, 0x0F, 0x0E, 0x00, 0x00, 0x70, 0xF0, 0x1E, 0x3C, 0x1F,
0xF0, 0x3C, 0x30, 0x18, 0x38, 0x7F, 0xFF, 0x1C, 0x38, 0x18, 0xF3, 0xF8, 0x2F, 0xCF, 0x1C, 0x38, 0xE7, 0xC0, 0x03, 0xE7, 0x1C,
0x33, 0xDE, 0x00, 0x00, 0x7B, 0xCC, 0x33, 0xBC, 0x00, 0x00, 0x3D, 0xCC, 0x3F, 0x78, 0x00, 0x00, 0x1E, 0xFC, 0x3F, 0x70,
0x08, 0x00, 0x0E, 0x7C, 0x3C, 0xE0, 0x1C, 0x00, 0x07, 0x3C, 0x39, 0xC0, 0x1C, 0x00, 0x03, 0x9C, 0x11, 0x80, 0x0C, 0x00, 0x01,
0x88, 0x03, 0x80, 0x0C, 0x00, 0x01, 0xC0, 0x03, 0x00, 0x0E, 0x00, 0x00, 0xC0, 0x07, 0x00, 0x06, 0x00, 0x00, 0xE0, 0x06, 0x00,
0x06, 0x00, 0x60, 0x0E, 0x00, 0x07, 0x00, 0x00, 0x60, 0x0E, 0x00, 0x03, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x03, 0x00, 0x00,
0x30, 0x0C, 0x00, 0x03, 0x80, 0x00, 0x30, 0x0C, 0x00, 0x7F, 0x80, 0x00, 0x30, 0x0C, 0x01, 0xFF, 0x80, 0x00, 0x30, 0x0C, 0x00,
0x55, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x00,
0x30, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x06, 0x00, 0x00, 0x00, 0x00, 0x60, 0x06, 0x00, 0x00, 0x00, 0x00, 0x60, 0x07, 0x00,
0x00, 0x00, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x03, 0x80, 0x00, 0x00, 0x01, 0xC0, 0x01, 0xC0, 0x00, 0x00, 0x03,
0x80, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x70, 0x00, 0x00, 0x0E, 0x00, 0x00, 0x78,
```

```
0x00, 0x00, 0x1E, 0x00, 0x00, 0xFE, 0x00, 0x00, 0x7F, 0x00, 0x01, 0xCF, 0x80, 0x01, 0xF3, 0x80, 0x03, 0x83, 0xF0, 0x1F, 0xC1,  
0xC0, 0x07, 0x00, 0xFF, 0xFF, 0x00, 0xE0, 0x06, 0x00, 0x1F, 0xF8, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00};
```

```
const unsigned char PROGMEM frame12 [] = {0x00, 0x00, 0x7E, 0x00, 0x00, 0x3F, 0x00, 0x03, 0xFF, 0x00, 0x00, 0xFF, 0xC0, 0x07, 0xCE, 0x00, 0x00, 0x79, 0xE0, 0xF, 0x1E, 0x00,
```

```
0x00, 0x78, 0xF0, 0x1C, 0x38, 0x1F, 0xF8, 0x1C, 0x38, 0x18, 0x78, 0xFF, 0xFF, 0x1E, 0x18, 0x38, 0xE3, 0xF0, 0x0F, 0xC7, 0x1C,
0x38, 0xCF, 0x80, 0x01, 0xF7, 0x8C, 0x33, 0xDE, 0x00, 0x00, 0x79, 0xCC, 0x37, 0xB8, 0x00, 0x00, 0x1D, 0xEC, 0x37, 0x38,
0x01, 0x80, 0x1E, 0xFC, 0x3E, 0xF0, 0x01, 0x80, 0x0E, 0x7C, 0x3C, 0xE0, 0x01, 0x80, 0x07, 0x3C, 0x39, 0xC0, 0x01, 0x80, 0x03,
0x9C, 0x13, 0x80, 0x01, 0x80, 0xC8, 0x03, 0x00, 0x01, 0x80, 0x00, 0xC0, 0x07, 0x00, 0x01, 0x80, 0x00, 0xE0, 0x07, 0x00,
0x01, 0x80, 0x00, 0x60, 0x06, 0x00, 0x01, 0x80, 0x00, 0x60, 0x06, 0x00, 0x01, 0x80, 0x00, 0x70, 0x0E, 0x00, 0x01, 0x80, 0x00,
0x70, 0x0C, 0x00, 0x01, 0x80, 0x00, 0x30, 0x0C, 0x00, 0x01, 0x80, 0x00, 0x30, 0x0C, 0x00, 0xFF, 0x80, 0x00, 0x30, 0x0C, 0x01,
0xFF, 0x80, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00,
0x30, 0x0C, 0x00, 0x00, 0x00, 0x30, 0x0E, 0x00, 0x00, 0x00, 0x70, 0x06, 0x00, 0x00, 0x00, 0x00, 0x60, 0x07, 0x00,
0x00, 0x00, 0x60, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x03, 0x80, 0x00, 0x00, 0x01, 0xC0, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x01, 0xC0, 0x00, 0x00, 0x01, 0x80, 0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x70,
0x00, 0x00, 0x0E, 0x00, 0x00, 0xFC, 0x00, 0x00, 0x3E, 0x00, 0x00, 0xFE, 0x00, 0x00, 0x7F, 0x80, 0x01, 0xC7, 0xC0, 0x03, 0xE3,
0x80, 0x03, 0x83, 0xFC, 0x1F, 0xC1, 0xC0, 0x07, 0x00, 0x7F, 0xFE, 0x00, 0xE0, 0x06, 0x00, 0x0F, 0xF0, 0x00, 0x60, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00};
```

```
0xFD, 0xC0, 0x01, 0x80, 0x03, 0x80, 0xFB, 0x80, 0x01, 0x80, 0x01, 0xC0, 0xF3, 0x80, 0x01, 0x80, 0x00, 0xC0, 0x67, 0x00, 0x01,
0x80, 0x00, 0xE0, 0x26, 0x00, 0x01, 0x80, 0x00, 0xE0, 0x06, 0x00, 0x01, 0x80, 0x00, 0x60, 0x0E, 0x00, 0x01, 0x80, 0x00, 0x60,
0x0E, 0x00, 0x01, 0x80, 0x00, 0x70, 0x0C, 0x00, 0x01, 0x80, 0x00, 0x30, 0x0C, 0x00, 0x01, 0x80, 0x00, 0x30, 0x0C, 0x00, 0xFF,
0x80, 0x00, 0x30, 0x0C, 0x01, 0xFF, 0x80, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30,
0x0C, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x06, 0x00, 0x00,
0x00, 0x00, 0x60, 0x07, 0x00, 0x00, 0x00, 0xE0, 0x07, 0x00, 0x00, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0,
0x03, 0x80, 0x00, 0x00, 0x01, 0xC0, 0x01, 0x80, 0x00, 0x00, 0x03, 0x80, 0x00, 0xE0, 0x00, 0x00, 0x03, 0x00, 0x00, 0xE0, 0x00,
0x00, 0x07, 0x00, 0x00, 0x70, 0x00, 0x0E, 0x00, 0x00, 0xFC, 0x00, 0x00, 0x3F, 0x00, 0x00, 0xFE, 0x00, 0x00, 0x7F, 0x00,
0x01, 0xC7, 0xC0, 0x01, 0xE3, 0x80, 0x03, 0x83, 0xF8, 0x2F, 0xC1, 0xC0, 0x07, 0x00, 0x7F, 0xFF, 0x00, 0xE0, 0x06, 0x00, 0x0F,
0xF0, 0x00, 0x60};
```

```

const unsigned char PROGMEM frame17 [] = {0x00, 0x3F, 0xC0, 0x00, 0x00, 0x00, 0x01, 0xFF, 0xE0, 0x00, 0x00, 0x00,
0x01, 0xE3, 0xE0, 0x00, 0x00, 0x03, 0x83, 0x80, 0x00, 0x1F, 0xC0, 0x07, 0x0F, 0x00, 0x00, 0x1F, 0xF0, 0x06, 0x1E, 0x1F,
0xF8, 0x0E, 0xF8, 0x0E, 0x78, 0xFF, 0x07, 0x1C, 0x0C, 0xF3, 0xF0, 0x0F, 0xC7, 0x0E, 0x0F, 0xEF, 0x80, 0x01, 0xF3, 0x86,
0x0F, 0x9E, 0x00, 0x00, 0x79, 0xC7, 0x0F, 0x38, 0x00, 0x00, 0x3C, 0xC7, 0x0E, 0x70, 0x01, 0x80, 0x0E, 0xE3, 0x04, 0x70, 0x01,
0x80, 0x0F, 0x73, 0x00, 0xE0, 0x01, 0x80, 0x07, 0x73, 0x01, 0xC0, 0x01, 0x80, 0x03, 0xBF, 0x03, 0x80, 0x01, 0x80, 0x01, 0xDF,
0x03, 0x00, 0x01, 0x80, 0x01, 0xCE, 0x07, 0x00, 0x01, 0x80, 0x00, 0xE6, 0x06, 0x00, 0x01, 0x80, 0x00, 0x64, 0x06, 0x00, 0x01,
0x80, 0x00, 0x60, 0x0E, 0x00, 0x01, 0x80, 0x00, 0x70, 0x0E, 0x00, 0x01, 0x80, 0x00, 0x70, 0x0C, 0x00, 0x01, 0x80, 0x00, 0x30,
0x0C, 0x00, 0x01, 0x80, 0x00, 0x30, 0x0C, 0x00, 0xFF, 0x80, 0x00, 0x30, 0x0C, 0x01, 0xFF, 0x80, 0x00, 0x30, 0x0C, 0x00, 0x00,
0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30,
0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x06, 0x00, 0x00, 0x00, 0x00, 0x60, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x03, 0x00, 0x00,
0x00, 0x00, 0xC0, 0x03, 0x80, 0x00, 0x00, 0x01, 0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0xC0, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80,
0x00, 0xC0, 0x00, 0x00, 0x07, 0x00, 0x00, 0xF0, 0x00, 0x07, 0x00, 0x00, 0x70, 0x00, 0x00, 0x0E, 0x00, 0x00, 0xFC, 0x00,
0x00, 0x3F, 0x00, 0x00, 0xFE, 0x00, 0x00, 0x7F, 0x00, 0x03, 0xC7, 0xC0, 0x03, 0xE3, 0x80, 0x03, 0x83, 0xF8, 0x3F, 0xC1, 0xE0,
0x07, 0x00, 0x7F, 0xFE, 0x00, 0xE0, 0x04, 0x00, 0x0F, 0xE0, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00};
```

```

const unsigned char PROGMEM frame21 [] = {0x00, 0x7F, 0xC0, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xE0, 0x00, 0x00, 0x00,
0x03, 0xE3, 0xE0, 0x00, 0x00, 0x03, 0x83, 0x80, 0x00, 0x1F, 0xC0, 0x07, 0x0F, 0x00, 0x00, 0x1F, 0xF0, 0x06, 0x1E, 0x3F,
0xF8, 0x0E, 0xF8, 0x0E, 0x78, 0xFF, 0xFF, 0x07, 0x1C, 0x0C, 0xF3, 0xF0, 0x0F, 0xC7, 0x0E, 0x0F, 0xEF, 0x80, 0x01, 0xF3, 0x86,
0x0F, 0x9E, 0x00, 0x00, 0x79, 0xC7, 0x0F, 0x38, 0x00, 0x00, 0x3C, 0xC7, 0x0E, 0x70, 0x00, 0x00, 0x0E, 0xE3, 0x0C, 0xF0, 0x00,
0x00, 0x0F, 0x73, 0x00, 0xE0, 0x00, 0x00, 0x07, 0x73, 0x01, 0xC0, 0x00, 0x00, 0x03, 0xBF, 0x01, 0x80, 0x00, 0x00, 0x01, 0xDF,
0x03, 0x80, 0x00, 0x00, 0x61, 0xCE, 0x07, 0x00, 0x00, 0x00, 0xE0, 0xE6, 0x06, 0x00, 0x00, 0x01, 0xC0, 0x64, 0x06, 0x00, 0x00,
0x07, 0x80, 0x60, 0x0E, 0x00, 0x00, 0x0F, 0x00, 0x70, 0x0E, 0x00, 0x00, 0x1E, 0x00, 0x70, 0x0C, 0x00, 0x00, 0x38, 0x00, 0x30,
0x0C, 0x00, 0x00, 0x70, 0x00, 0x30, 0x0C, 0x00, 0xFF, 0xE0, 0x00, 0x30, 0x0C, 0x01, 0xFF, 0x80, 0x00, 0x30, 0x0C, 0x00, 0x21,
0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30,
0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x06, 0x00, 0x00, 0x00, 0x00, 0x60, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x03, 0x00, 0x00,
0x00, 0x00, 0xE0, 0x03, 0x80, 0x00, 0x00, 0x01, 0xC0, 0x03, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80,
```

```
0x00, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x00, 0xF0, 0x00, 0x00, 0xF, 0x00, 0x00, 0x70, 0x00, 0x00, 0xE, 0x00, 0x00, 0x7C, 0x00,
0x00, 0x3E, 0x00, 0x01, 0xFE, 0x00, 0x00, 0x7F, 0x80, 0x01, 0xC7, 0xC0, 0x03, 0xE3, 0x80, 0x07, 0x83, 0xF8, 0x3F, 0xC1, 0xE0,
0x07, 0x00, 0x7F, 0xFE, 0x00, 0xE0, 0x04, 0x00, 0x0F, 0xE8, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00};
```

```

const unsigned char PROGMEM frame22 [] = {0x00, 0x3F, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xC0, 0x00, 0x00, 0x00,
0x03, 0xE7, 0xC0, 0x00, 0x1F, 0x00, 0x07, 0x87, 0x00, 0x00, 0x3F, 0xE0, 0x06, 0x0F, 0x00, 0x00, 0x1F, 0xF0, 0x0E, 0x1C, 0x3F,
0xF8, 0x1C, 0x78, 0x0C, 0x38, 0xFF, 0xFF, 0x0E, 0x1C, 0x0C, 0xF3, 0xF0, 0x0F, 0xC7, 0x0C, 0x19, 0xEF, 0x80, 0x01, 0xF3,
0x8E, 0x1F, 0xDE, 0x00, 0x00, 0x79, 0xC6, 0x1F, 0x38, 0x00, 0x00, 0x3C, 0xE6, 0x1E, 0x70, 0x00, 0x00, 0x0E, 0xE6, 0x1E, 0xF0,
0x00, 0x00, 0x07, 0x76, 0x08, 0xE0, 0x00, 0x00, 0x07, 0x7E, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x9E, 0x03, 0x80, 0x00, 0x00, 0x01,
0xDE, 0x03, 0x00, 0x00, 0x00, 0x01, 0xCC, 0x07, 0x00, 0x00, 0x00, 0xE4, 0x06, 0x00, 0x00, 0x00, 0x00, 0x60, 0x06, 0x00,
0x00, 0x00, 0x00, 0x60, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x70, 0xC0, 0x00, 0x00, 0x00,
0x1C, 0x30, 0x0C, 0x00, 0x00, 0x07, 0xFC, 0x30, 0x0C, 0x00, 0xFF, 0xFF, 0xC0, 0x30, 0x0C, 0x01, 0xFF, 0xE8, 0x00, 0x30, 0x0C, 0x00,
0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x30, 0x0E, 0x00,
0x00, 0x00, 0x00, 0x00, 0x70, 0x06, 0x00, 0x00, 0x00, 0x00, 0x60, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x03, 0x00,
0x00, 0x00, 0xC0, 0x03, 0x80, 0x00, 0x00, 0x01, 0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0xC0, 0x01, 0xC0, 0x00, 0x00, 0x03,
0x80, 0x00, 0xE0, 0x00, 0x03, 0x00, 0x00, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x0E, 0x00, 0x00,
0x7C, 0x00, 0x00, 0x3E, 0x00, 0x01, 0xFE, 0x00, 0x00, 0xFF, 0x80, 0x01, 0xC7, 0xC0, 0x03, 0xE3, 0x80, 0x07, 0x83, 0xFA, 0x3F, 0xC1,
0xE0, 0x07, 0x00, 0x7F, 0xFE, 0x00, 0xE0, 0x04, 0x00, 0x0F, 0xE0, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00};
```

```

const unsigned char PROGMEM frame23 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x3C, 0x00,
0x01, 0xFF, 0x00, 0x00, 0xFF, 0x80, 0x07, 0xEF, 0x00, 0x00, 0xFB, 0xE0, 0x0F, 0x0E, 0x00, 0x00, 0x70, 0xF0, 0x1E, 0x3C, 0x1F,
0xF0, 0x3C, 0x38, 0x18, 0x38, 0x7F, 0xFF, 0x1C, 0x38, 0x18, 0xF3, 0xF8, 0x2F, 0xCF, 0x1C, 0x38, 0xE7, 0xC0, 0x03, 0xE7, 0x1C,
0x33, 0xDE, 0x00, 0x00, 0x7B, 0xCC, 0x33, 0xBC, 0x00, 0x00, 0x3D, 0xCC, 0x3F, 0x78, 0x00, 0x00, 0x1E, 0xFC, 0x3F, 0x70,
0x00, 0x00, 0x0E, 0x7C, 0x3C, 0xE0, 0x00, 0x00, 0x07, 0x3C, 0x39, 0xC0, 0x00, 0x00, 0x03, 0x9C, 0x11, 0x80, 0x00, 0x00, 0x01,
0x88, 0x03, 0x80, 0x00, 0x00, 0x01, 0xC0, 0x03, 0x00, 0x00, 0x00, 0xE0, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x06, 0x00,
0x00, 0x00, 0x00, 0x60, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x60, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x00, 0x00,
0x30, 0x0C, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x7F, 0x80, 0x00, 0x30, 0x0C, 0x01, 0xFF, 0xE0, 0x00, 0x30, 0x0C, 0x00,
0xAA, 0xF8, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x3E, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x0F, 0xC0, 0x30, 0x0C, 0x00, 0x00, 0x03, 0xF0,
0x30, 0x0E, 0x00, 0x00, 0x00, 0x78, 0x70, 0x06, 0x00, 0x00, 0x00, 0x18, 0x60, 0x06, 0x00, 0x00, 0x00, 0x00, 0x60, 0x07, 0x00,
0x00, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00, 0xC0, 0x03, 0x80, 0x00, 0x00, 0x01, 0xC0, 0x01, 0xC0, 0x00, 0x00, 0x03,
0x80, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x70, 0x00, 0x00, 0x0E, 0x00, 0x00, 0x78,
0x00, 0x00, 0x1E, 0x00, 0x00, 0xFE, 0x00, 0x00, 0x7F, 0x00, 0x01, 0xCF, 0x80, 0x01, 0xF3, 0x80, 0x03, 0x83, 0xF0, 0x1F, 0xC1,
0xC0, 0x07, 0x00, 0xFF, 0xFF, 0x00, 0xE0, 0x06, 0x00, 0x1F, 0xF8, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00};
```

```

const unsigned char PROGMEM frame24 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x3C, 0x00,
0x01, 0xFF, 0x00, 0x00, 0xFF, 0x80, 0x07, 0xEF, 0x00, 0x00, 0xFB, 0xE0, 0x0F, 0x0E, 0x00, 0x00, 0x70, 0xF0, 0x1E, 0x3C, 0x1F,
0xF0, 0x3C, 0x38, 0x18, 0x38, 0x7F, 0xFF, 0x1C, 0x38, 0x18, 0xF3, 0xF8, 0x2F, 0xCF, 0x1C, 0x38, 0xE7, 0xC0, 0x03, 0xE7, 0x1C,
0x33, 0xDE, 0x00, 0x00, 0x7B, 0xCC, 0x33, 0xBC, 0x00, 0x00, 0x3D, 0xCC, 0x3F, 0x78, 0x00, 0x00, 0x1E, 0xFC, 0x3F, 0x70,
0x00, 0x00, 0x0E, 0x7C, 0x3C, 0xE0, 0x00, 0x00, 0x07, 0x3C, 0x39, 0xC0, 0x00, 0x00, 0x03, 0x9C, 0x11, 0x80, 0x00, 0x00, 0x01,
0x88, 0x03, 0x80, 0x00, 0x00, 0x01, 0xC0, 0x03, 0x00, 0x00, 0x00, 0xE0, 0x07, 0x00, 0x00, 0x00, 0xE0, 0x06, 0x00,
0x00, 0x00, 0x00, 0x60, 0x0E, 0x00, 0x00, 0x00, 0x60, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x00, 0x00,
0x30, 0x0C, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x7F, 0x80, 0x00, 0x30, 0x0C, 0x01, 0xFF, 0x80, 0x00, 0x30, 0x0C, 0x00,
0x55, 0xC0, 0x00, 0x30, 0x0C, 0x00, 0x00, 0xC0, 0x00, 0x30, 0x0C, 0x00, 0x00, 0xE0, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x60, 0x00,
0x30, 0x0E, 0x00, 0x00, 0x70, 0x00, 0x70, 0x06, 0x00, 0x00, 0x38, 0x00, 0x60, 0x06, 0x00, 0x00, 0x38, 0x00, 0x60, 0x07, 0x00,
0x00, 0x1C, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x0C, 0x00, 0xC0, 0x03, 0x80, 0x00, 0x0E, 0x01, 0xC0, 0x01, 0xC0, 0x00, 0x0E, 0x03,
0x80, 0x01, 0xC0, 0x00, 0x06, 0x03, 0x80, 0x00, 0xE0, 0x00, 0x02, 0x07, 0x00, 0x00, 0x70, 0x00, 0x00, 0x0E, 0x00, 0x00, 0x78,
0x00, 0x00, 0x1E, 0x00, 0x00, 0xFE, 0x00, 0x00, 0x7F, 0x00, 0x01, 0xCF, 0x80, 0x01, 0xF3, 0x80, 0x03, 0x83, 0xF0, 0x1F, 0xC1,
```

```

const unsigned char PROGMEM frame26 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x3C, 0x00,
0x01, 0xFF, 0x00, 0x00, 0xFF, 0x80, 0x07, 0xEF, 0x00, 0x00, 0xFB, 0xE0, 0x0F, 0x0E, 0x00, 0x00, 0x70, 0xF0, 0x1E, 0x3C, 0x1F,
0xF0, 0x3C, 0x38, 0x18, 0x38, 0x7F, 0xFF, 0x1C, 0x38, 0x18, 0xF3, 0xF8, 0x2F, 0xCF, 0x1C, 0x38, 0xE7, 0xC0, 0x03, 0xE7, 0x1C,
0x33, 0xDE, 0x00, 0x00, 0x7B, 0xCC, 0x33, 0xBC, 0x00, 0x00, 0x3D, 0xCC, 0x3F, 0x78, 0x00, 0x00, 0x1E, 0xFC, 0x3F, 0x70,
0x00, 0x00, 0x0E, 0x7C, 0x3C, 0xE0, 0x00, 0x00, 0x07, 0x3C, 0x39, 0xC0, 0x00, 0x00, 0x03, 0x9C, 0x11, 0x80, 0x00, 0x00, 0x01,
0x88, 0x03, 0x80, 0x00, 0x00, 0x01, 0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x06, 0x00,
0x00, 0x00, 0x00, 0x60, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x60, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x00, 0x00,
0x30, 0x0C, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0xFF, 0x80, 0x00, 0x30, 0x0C, 0x01, 0xFF, 0x80, 0x00, 0x30, 0x0C, 0x00,
0x5F, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x1E, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x38, 0x00, 0x00, 0x30, 0x0C, 0x00, 0xF0, 0x00, 0x00,
0x30, 0x0E, 0x01, 0xE0, 0x00, 0x00, 0x70, 0x06, 0x07, 0x80, 0x00, 0x00, 0x60, 0x06, 0x0F, 0x00, 0x00, 0x00, 0x60, 0x07, 0x0C,
0x00, 0x00, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00, 0xC0, 0x03, 0x80, 0x00, 0x00, 0x01, 0xC0, 0x01, 0xC0, 0x00, 0x00, 0x03,
0x80, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x70, 0x00, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xFE, 0x00, 0x00, 0x7F, 0x00, 0x01, 0xCF, 0x80, 0x01, 0xF3, 0x80, 0x03, 0x83, 0xF0, 0x1F, 0xC1,
0xC0, 0x07, 0x00, 0xFF, 0xFF, 0x00, 0xE0, 0x06, 0x00, 0x1F, 0xF8, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00};
```

```

const unsigned char PROGMEM frame27 [] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x3C, 0x00,
0x01, 0xFF, 0x00, 0x00, 0xFF, 0x80, 0x07, 0xEF, 0x00, 0x00, 0xFB, 0xE0, 0x0F, 0xE0, 0x00, 0x00, 0x70, 0xF0, 0x1E, 0x3C, 0x1F,
0xF0, 0x3C, 0x30, 0x18, 0x38, 0x7F, 0xFF, 0x1C, 0x38, 0x18, 0xF3, 0xF8, 0x2F, 0xCF, 0x1C, 0x38, 0xE7, 0xC0, 0x03, 0xE7, 0x1C,
0x33, 0xDE, 0x00, 0x00, 0x7B, 0xCC, 0x33, 0xBC, 0x00, 0x00, 0x3D, 0xCC, 0x3F, 0x78, 0x00, 0x00, 0x1E, 0xFC, 0x3F, 0x70,
0x00, 0x00, 0x0E, 0x7C, 0x3C, 0xE0, 0x00, 0x00, 0x07, 0x3C, 0x39, 0xC0, 0x00, 0x00, 0x03, 0x9C, 0x11, 0x80, 0x00, 0x00, 0x01,
0x88, 0x03, 0x80, 0x00, 0x00, 0x01, 0xC0, 0x03, 0x00, 0x00, 0x00, 0xC0, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x06, 0x00,
0x00, 0x00, 0x00, 0x60, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x60, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x00, 0x00,
0x30, 0x0C, 0x38, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x3F, 0xFF, 0x80, 0x00, 0x30, 0x0C, 0x03, 0xFF, 0x80, 0x00, 0x30, 0x0C, 0x00,
0x55, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x00,
0x30, 0x0E, 0x00, 0x00, 0x00, 0x70, 0x06, 0x00, 0x00, 0x00, 0x00, 0x60, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x60, 0x07, 0x00,
0x00, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00, 0xC0, 0x03, 0x80, 0x00, 0x00, 0x01, 0xC0, 0x01, 0xC0, 0x00, 0x00, 0x03,
0x80, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x70, 0x00, 0x00, 0x0E, 0x00, 0x00, 0x78,
0x00, 0x00, 0x1E, 0x00, 0x00, 0xFE, 0x00, 0x00, 0x7F, 0x00, 0x01, 0xCF, 0x80, 0x01, 0xF3, 0x80, 0x03, 0x83, 0xF0, 0x1F, 0xC1,
0xC0, 0x07, 0x00, 0xFF, 0xFF, 0x00, 0xE0, 0x06, 0x00, 0x1F, 0xF8, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00};
```



```
0x86, 0x13, 0x00, 0x00, 0xD8, 0x61, 0x86, 0x3B, 0x00, 0x00, 0xCC, 0x61, 0x86, 0x13, 0x00, 0x00, 0xDD, 0x75, 0xAE, 0xBB,  
0x00, 0x00, 0xCF, 0xFF, 0xFF, 0xF3, 0x00, 0x00, 0xDE, 0xFB, 0xDF, 0x7B, 0x00, 0x00, 0xCC, 0x61, 0x86, 0x13, 0x00, 0x00,  
0xD8, 0x61, 0x86, 0x3B, 0x00, 0x00, 0xCC, 0x61, 0x86, 0x13, 0x00, 0x00, 0xDF, 0xFF, 0xFF, 0xFB, 0x00, 0x00, 0xCF, 0xFF,  
0xFF, 0xF3, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x03, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x03, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00,  
0x00, 0x7F, 0xFF, 0xFF, 0xFE, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
```


|||||||||||||||||||||||LUPA


```
const unsigned char PROGMEM lupa10 [] = {0x00, 0x00, 0x00, 0x05, 0x80, 0x00, 0x00, 0x00, 0x00, 0x3F, 0xF8, 0x00,
0x00, 0x00, 0x00, 0xF8, 0x3C, 0x00, 0x00, 0x01, 0xC0, 0x07, 0x00, 0x00, 0x00, 0x03, 0x00, 0x03, 0x80, 0x00, 0x00, 0x06,
0x00, 0x00, 0xC0, 0x00, 0x00, 0x0C, 0x00, 0x00, 0x60, 0x00, 0x00, 0x08, 0x00, 0x00, 0x60, 0x00, 0x00, 0x18, 0x00, 0x00, 0x30,
0x00, 0x00, 0x30, 0x00, 0x00, 0x10, 0x00, 0x00, 0x30, 0x00, 0x00, 0x18, 0x00, 0x00, 0x30, 0x00, 0x00, 0x18, 0x00, 0x00, 0x60,
0x00, 0x00, 0x18, 0x00, 0x00, 0x20, 0x00, 0x00, 0x08, 0x00, 0x00, 0x60, 0x00, 0x00, 0x18, 0x00, 0x00, 0x20, 0x00, 0x00, 0x08,
0x00, 0x00, 0x20, 0x00, 0x00, 0x18, 0x00, 0x00, 0x30, 0x00, 0x00, 0x18, 0x00, 0x00, 0x30, 0x00, 0x00, 0x10, 0x00, 0x00, 0x10,
0x00, 0x00, 0x30, 0x00, 0x00, 0x18, 0x00, 0x00, 0x30, 0x00, 0x00, 0x08, 0x00, 0x00, 0x60, 0x00, 0x00, 0x0E, 0x00, 0x00, 0x0C0,
0x00, 0x00, 0x06, 0x00, 0x01, 0x80, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x03, 0xC0, 0x0E, 0x00, 0x00, 0x00, 0x07,
0xFD, 0x7C, 0x00, 0x00, 0x00, 0x06, 0xDF, 0xE0, 0x00, 0x00, 0x00, 0x0F, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1D, 0x80, 0x00, 0x00,
0x00, 0x00, 0x3B, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x66, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7C,
0x00, 0x00, 0x00, 0x00, 0x00, 0xEC, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xB8, 0x00, 0x00, 0x00, 0x00, 0x01, 0xD8, 0x00, 0x00, 0x00,
0x00, 0x03, 0x70, 0x00, 0x00, 0x00, 0x00, 0x07, 0xB0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0E, 0x00,
0x00, 0x00, 0x00, 0x00, 0x1D, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x1D, 0x80, 0x00, 0x00, 0x00, 0x00, 0x1B, 0x80, 0x00, 0x00, 0x00,
0x00, 0x1F, 0x00, 0x04, 0x00, 0x00};
```

```
const unsigned char PROGMEM lupa11 [] = {0x00, 0x00, 0x00, 0x0B, 0x80, 0x00, 0x00, 0x00, 0x00, 0x7F, 0xF0, 0x00,
0x00, 0x00, 0xF0, 0x3C, 0x00, 0x00, 0x03, 0x80, 0x0F, 0x00, 0x00, 0x00, 0x07, 0x00, 0x03, 0x00, 0x00, 0x00, 0x0C,
0x00, 0x01, 0xC0, 0x00, 0x00, 0x1C, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x18, 0x00, 0x00, 0x60, 0x00, 0x00, 0x00, 0x30, 0x00, 0x00, 0x20,
0x00, 0x00, 0x30, 0x00, 0x00, 0x30, 0x00, 0x00, 0x20, 0x00, 0x00, 0x30, 0x00, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10,
0x00, 0x00, 0x18, 0x00, 0x00, 0x40, 0x00, 0x00, 0x10, 0x00, 0x00, 0x60, 0x00, 0x00, 0x18, 0x00, 0x00, 0x60, 0x00, 0x00, 0x00, 0x18,
0x00, 0x00, 0x60, 0x00, 0x00, 0x18, 0x00, 0x00, 0x30, 0x00, 0x00, 0x10, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x30, 0x00, 0x00, 0x00, 0x30,
0x00, 0x00, 0x30, 0x00, 0x00, 0x18, 0x00, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x0C0,
0x00, 0x00, 0x06, 0x00, 0x01, 0x80, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x03, 0xC0, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F,
0xFA, 0xF8, 0x00, 0x00, 0x00, 0x0D, 0x9F, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x1F, 0x80, 0x00, 0x00, 0x00, 0x00, 0x3B, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x36, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xEC, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xB8,
0x00, 0x00, 0x00, 0x00, 0x01, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xB0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0xF0, 0x00, 0x00, 0x00, 0x00,
0x00, 0x07, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0xE0, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1D, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1D, 0x80,
0x00, 0x00, 0x00, 0x00, 0x3B, 0x80, 0x00, 0x00, 0x00, 0x00, 0x37, 0x00, 0x00, 0x00, 0x00, 0x00, 0x77, 0x00, 0x00};
```

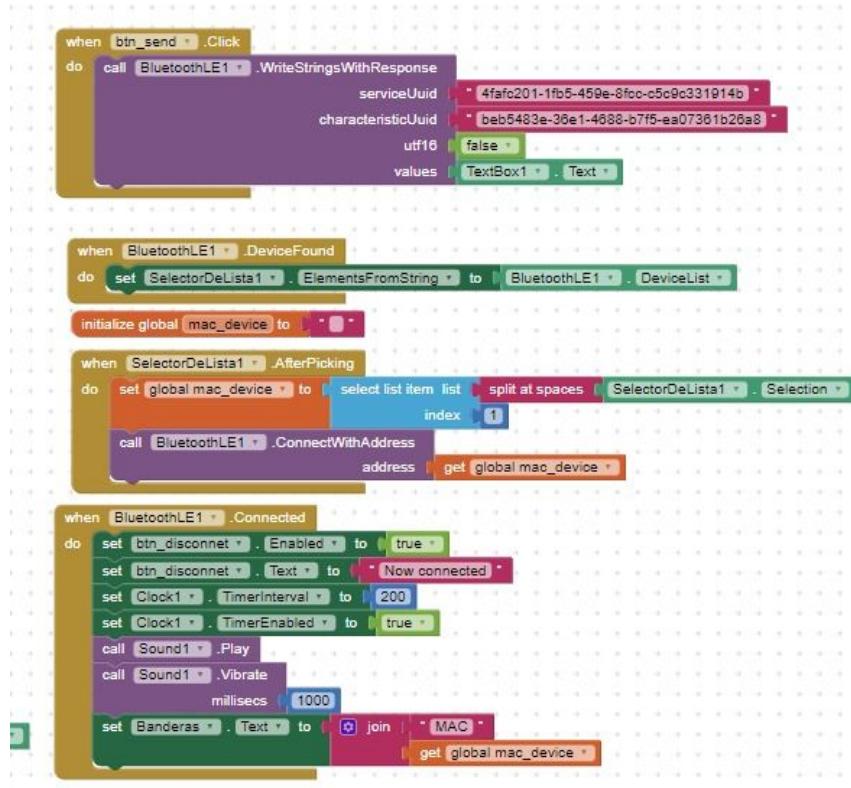
```
const unsigned char PROGMEM lupa12 [] = {0x00, 0x00, 0x00, 0x0B, 0x80, 0x00, 0x00, 0x00, 0x00, 0x7F, 0xF0, 0x00,
0x00, 0x00, 0xF0, 0x3C, 0x00, 0x00, 0x03, 0x80, 0x0E, 0x00, 0x00, 0x00, 0x06, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00, 0x00, 0x0C,
0x00, 0x01, 0x80, 0x00, 0x00, 0x18, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x30, 0x00, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x30, 0x00, 0x00, 0x60,
0x00, 0x00, 0x60, 0x00, 0x00, 0x20, 0x00, 0x00, 0x60, 0x00, 0x00, 0x30, 0x00, 0x00, 0x60, 0x00, 0x00, 0x00, 0x30, 0x00, 0x00, 0x00, 0xC0,
0x00, 0x00, 0x10, 0x00, 0x00, 0x40, 0x00, 0x00, 0x30, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x40, 0x00, 0x00, 0x30,
0x00, 0x00, 0x40, 0x00, 0x00, 0x10, 0x00, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20,
0x00, 0x00, 0x60, 0x00, 0x00, 0x30, 0x00, 0x00, 0x60, 0x00, 0x00, 0x18, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x01, 0x80,
0x00, 0x00, 0x0E, 0x00, 0x03, 0x00, 0x00, 0x00, 0x0F, 0x00, 0x07, 0x00, 0x00, 0x00, 0x00, 0x0F, 0xC0, 0x1C, 0x00, 0x00, 0x00, 0x1B,
0xFA, 0xF8, 0x00, 0x00, 0x00, 0x37, 0x1F, 0xC0, 0x00, 0x00, 0x00, 0x3F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0xDC, 0x00, 0x00, 0x00, 0x01, 0xDC, 0x00, 0x00, 0x00, 0x00, 0x01, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xB0,
0x00, 0x00, 0x00, 0x00, 0x07, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0xE0, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1D, 0xC0, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1D, 0xC0, 0x00, 0x00, 0x00, 0x3B, 0x80, 0x00, 0x00, 0x00, 0x37, 0x00, 0x00, 0x00, 0x00, 0x00, 0x77, 0x00, 0x00};
```

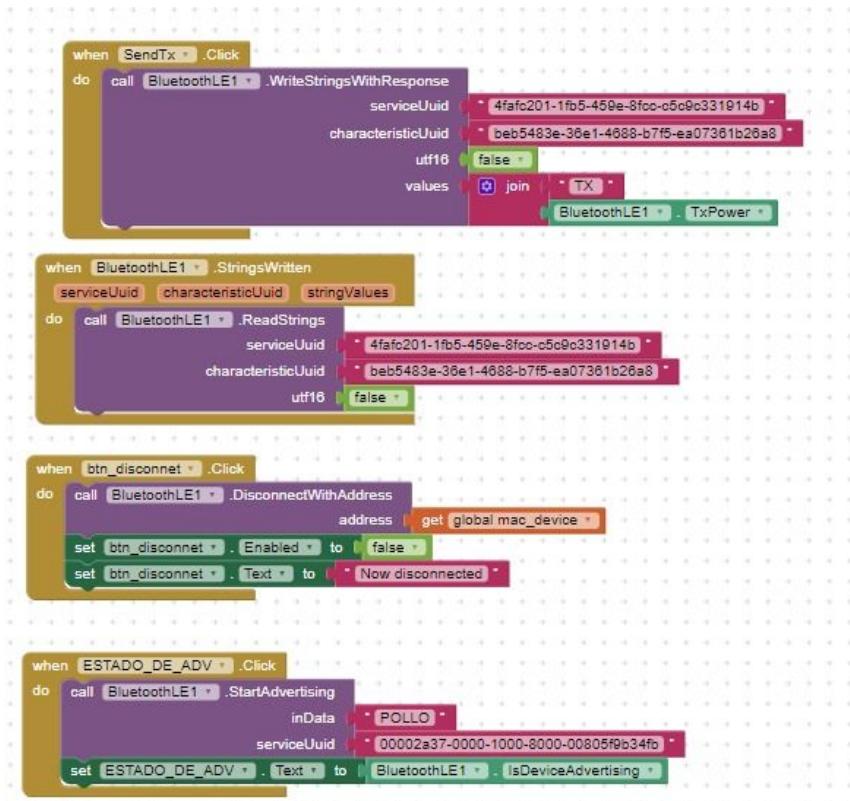
```
const unsigned char PROGMEM lupa13 [] = {0x00, 0x00, 0x00, 0x0B, 0x80, 0x00, 0x00, 0x00, 0x00, 0x7F, 0xE0, 0x00,
0x00, 0x00, 0xF0, 0x78, 0x00, 0x00, 0x00, 0x07, 0x00, 0x1E, 0x00, 0x00, 0x00, 0x0E, 0x00, 0x06, 0x00, 0x00, 0x00, 0x00, 0x0C,
0x00, 0x03, 0x80, 0x00, 0x00, 0x18, 0x00, 0x01, 0x80, 0x00, 0x00, 0x00, 0x30, 0x00, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x30, 0x00, 0x00, 0x60,
```

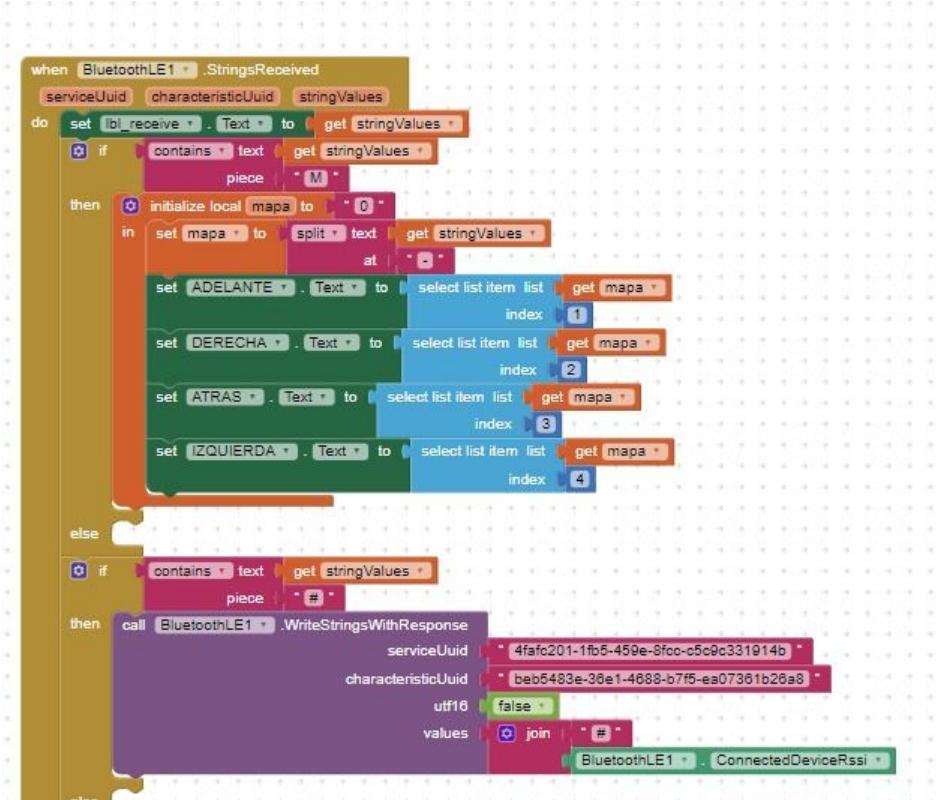


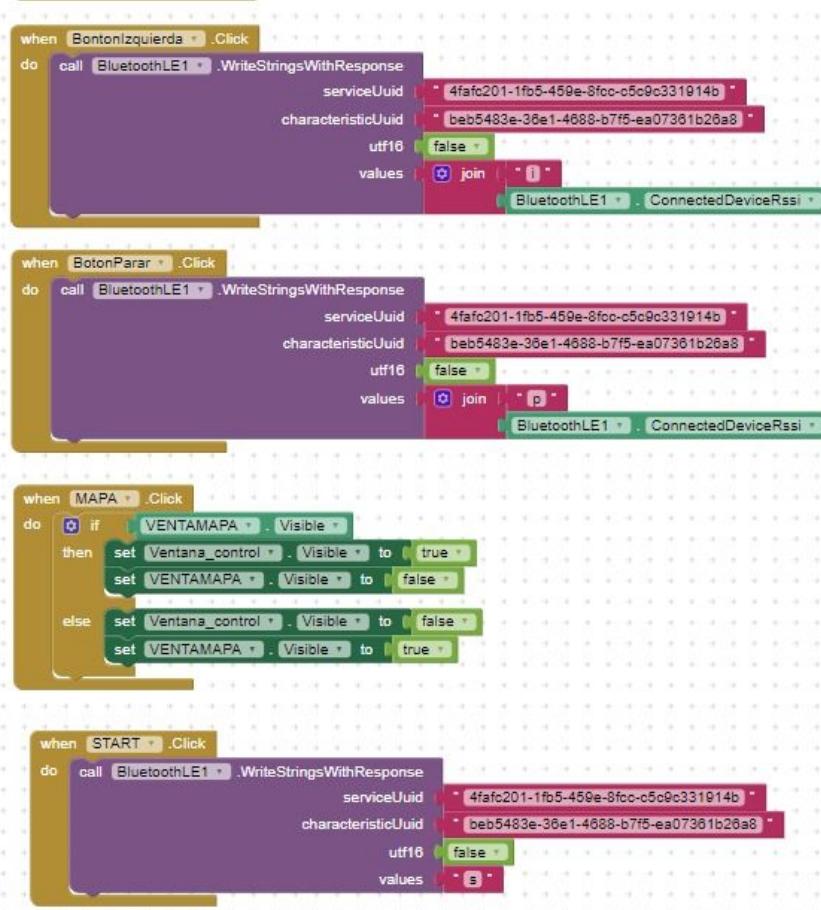
```
//const unsigned char PROGMEM telefono[] = {phone0, phone1, phone2, phone3, phone4, phone5, phone6, phone7,  
phone8, phone9, phone10, phone11, phone12, phone13, phone14, phone15, phone16, phone17, phone18, phone19, phone20, phone21,  
phone22, phone23, phone24, phone25, phone26, phone27};  
#endif
```

9.4.3. APP Mit Inventor









```

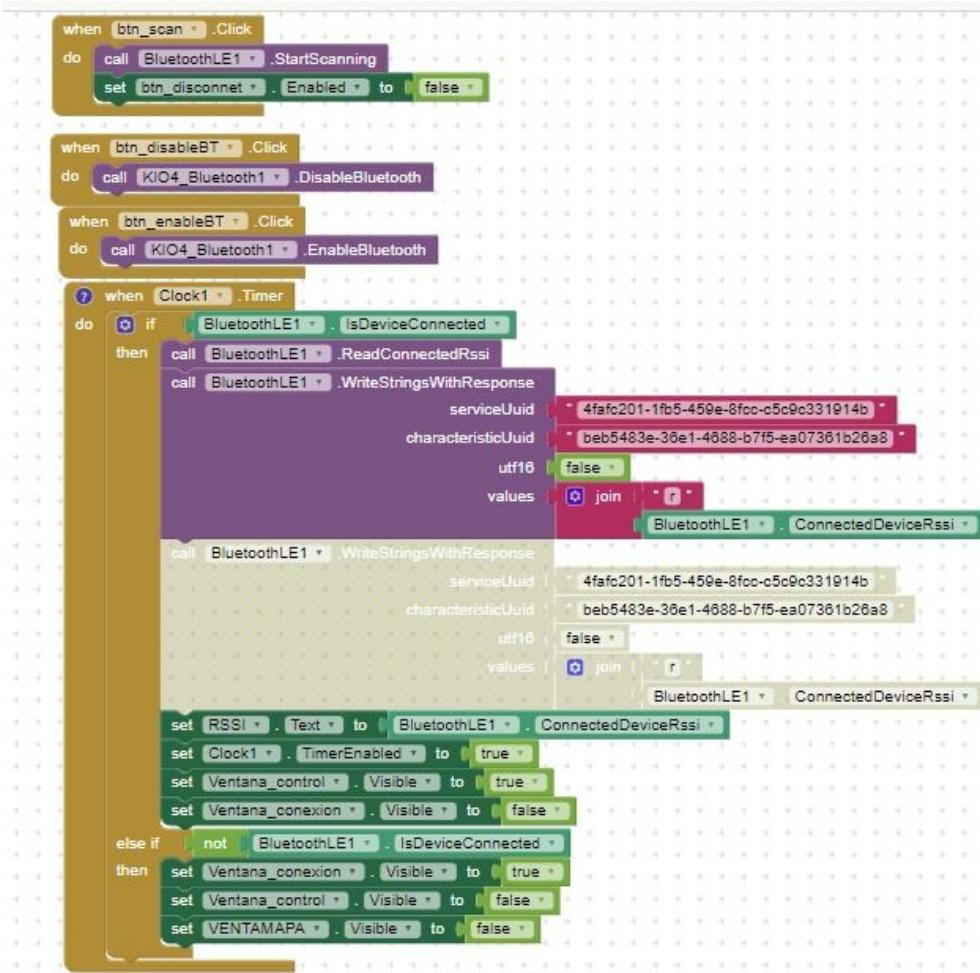
when [RSSI v].Click
do set RSSI v Text to BluetoothLE1 . ConnectedDeviceRssi

when [Calibrar v].Click
do call BluetoothLE1 .WriteStringsWithResponse
    serviceUuid "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
    characteristicUuid "beb5483e-36e1-4688-b7f5-ea07361b26a8"
    utf16 false
    values [join ["c"]]
    BluetoothLE1 . ConnectedDeviceRssi

when [BotonAdelante v].Click
do call BluetoothLE1 .WriteStringsWithResponse
    serviceUuid "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
    characteristicUuid "beb5483e-36e1-4688-b7f5-ea07361b26a8"
    utf16 false
    values [join ["a"]]
    BluetoothLE1 . ConnectedDeviceRssi

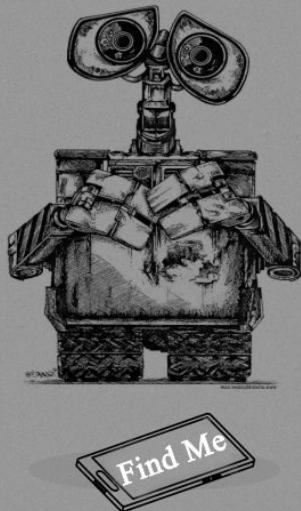
when [BotonAtras v].Click
do call BluetoothLE1 .WriteStringsWithResponse
    serviceUuid "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
    characteristicUuid "beb5483e-36e1-4688-b7f5-ea07361b26a8"
    utf16 false
    values [join ["b"]]
    BluetoothLE1 . ConnectedDeviceRssi

when [BotonDerecha v].Click
do call BluetoothLE1 .WriteStringsWithResponse
    serviceUuid "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
    characteristicUuid "beb5483e-36e1-4688-b7f5-ea07361b26a8"
    utf16 false
    values [join ["x"]]
    BluetoothLE1 . ConnectedDeviceRssi
  
```



9.5. POSTER

Hide & Seek



El robot que te sigue a todos lados



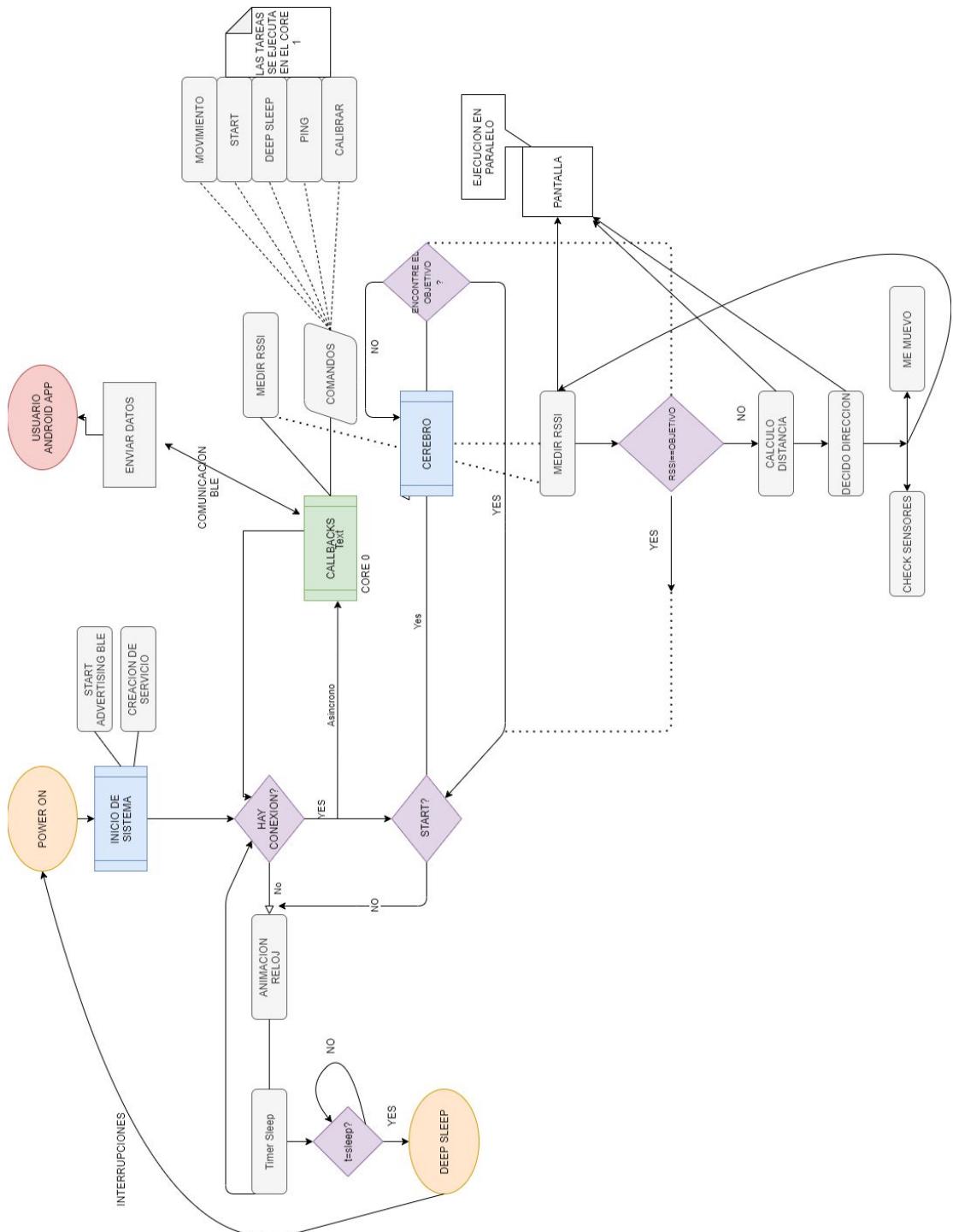
Cualquier parecido con la pelicula es pura coincidencia

9.6. Comandos

1. PING == '#'
2. Led off== "off"
3. Mostrar datos en serial=="show") { //esto puede ser moverse adelante por ejemplo
4. Led == "on") {
5. Calibrar == 'c'+”rssI”
6. Deep sleep == 'd'
7. Mover adelante == 'a'
8. parar == 'p'
9. Girar izquierda == 'i'
10. Girar a la derecha == 'x'
11. Mover atras == 'b'
12. Start, inicair busqueda == "s"
13. Calibrar giro == 't'+”tiempo”
14. Dif de PWM de las ruedas en el giro== 'k'+”cantidad”
15. Tiempo extra en cerebro == 'z')

```
16.    pCharacteristic->setValue("tiempo para pensar");
17.    esperoExtra();
18.    Dif de PWM en las ruedas de adelante == '!'
19.    Probar otra eq de distancia == 'e' {
```

9.7. Diagrama de flujo



9.8 Link Al video

https://drive.google.com/file/d/1aDOSe8tAoxmj_YXvfb8OhieH5EbsroJ/view?usp=sharing

9.9 Link A Poster

<https://drive.google.com/drive/folders/1Mag6ShrnBWlaQZ4cR82-JaoUdArvARST?usp=sharing>