“The watcher”

UNIVERSIDAD AUTÓNOMA DE CHIHUAHUA

FACULTAD DE INGENIERÍA, ISCH.

MAESTRO: Alberto Pacheco González

CLASE: Aplicaciones embebidas para internet de las cosas

ALUMNOS:

320631 - Brandon Saldívar Trevizo

325103 - Sergio Alan Aceves Chávez

329625 – Daniel Rascón Carrillo

25 de mayo de 2022

Contenido

Planteamiento del Proyecto

Debido al alta en inseguridad que ha surgido en el país en el que vivimos (México), el equipo pretende atender esta problemática, implementando un dispositivo de IoT nombrado como “The watcher” que sea capaz (mediante programación en el lenguaje C++) de capturar video con un constante traque a la fuente que produzca algún sonido en particular, actuando tal que esta pueda ser posicionada y permita mediante el video-streaming poder fungir como cámara de seguridad.

Objetivos

Con este proyecto pretendemos principalmente denotar y a su vez pulir nuestras habilidades obtenidas a lo largo de la carrera de ingeniería en sistemas computacionales en hardware, específicamente, en los ámbitos de programación (siendo C++ nuestro lenguaje designado para el proyecto), y la electrónica, pues con este pretendemos dar un enfoque laboral lo más acercado al rubro comercial y laboral posible.

De igual manera pretendemos utilizar esta experiencia con el fin de la acreditación de la clase de Aplicaciones embebidas para internet de las cosas, impartida por el docente Alberto Pacheco.

Metodología

Obviamente en el proyecto encontramos un sin número de prototipos, más que nada en cuestiones de programación, donde tuvimos que realizar fuertes testeos antes de poder culminar en el dispositivo que buscábamos.

Es por eso que para nuestro proyecto fue destinada al planteamiento de prototipos, ya que la misma se adecuo de manera excepcional al planteamiento con el que se pretendía trabajar, ya que, al encontrarnos actualmente en pandemia, nos permite trabajar con distintas fases del proyecto de manera remota, para de esta manera

posteriormente consolidarlas, dándonos como culminación una serie de prototipos que nos permiten entender de mejor manera el rendimiento y la vida de nuestro proyecto.

Por estas razones y por como lo dice su definición "*La metodología de prototipado está relacionada con la mejora continua y el Ciclo de Deming que consiste en un proceso iterativo enfocado en diseñar, implementar, medir y ajustar un plan.*" Fue que continuamos realizando múltiples pruebas en busca de la mejora continua.

Descripción de materiales

En el presente apartado denotaremos el material utilizado para el desarrollo del proyecto en cuestión, de igual manera nombraremos algunos atributos específicos de algunos componentes que delimitaran nuestra elección por los mismos ya que siendo de nuestro conocimiento que existe más productos compitiendo en el mismo mercado con prestaciones similares pretendemos encontrar la mejor opción para el proyecto en cuestión.

ESP32

El ESP32 es un micro controlador desarrollado por Espressif Systems, capaz de proporcionarnos en un mismo chip un sistema con Wi-Fi y – Bluethooth que recuerda a un Arduino con capacidades mayores, permitiendo ser una opción barata y con mejores prestaciones que el anteriormente nombrado, se ha optado por este controlador, ya que dentro de las muchas ventajas que este nos ofrece el poder contar con soporte para lenguaje de programación en C++ y una gran cantidad de pines a disposición, lo convirtió en una opción idónea para el proyecto.

Imagen tomada de un ESP32 con soporte Wifi/Bluethoot

ESP32-CAM

Al trabajar con el ESP32 la versión “CAM” nos brinda además de múltiples pines GPIO una cámara de video, así como una ranura para tarjetas MicroSD, que aunado a las ventajas que antes mencionamos del Wi-Fi y Bluethoot, nos permiten hacer streamaing del video captado en el mismo mostrándonos una calidad aceptable para un dispositivo tan pequeño, con posibles funciones para seguimiento, e incluso reconocimiento facial.

Imagen de un ESP32-CAM

Micrófono KY-037

El módulo KY-037 funge como micrófono, el cual permite detectar sonido mediante su señal analógica, usando las variaciones en función de las frecuencias que estas producen, el dispositivo también cuenta con un trimpot, que no es nada más y nada menos que un potenciómetro que nos permite configurar la sensibilidad del mismo, dándonos la oportunidad de captar sonidos menos fuertes; este módulo también puede ser usado con señales digitales, sin embargo, no serán requeridas para este proyecto.

Especificaciones y características

- Peso: 4g.
- Voltaje: 3,3 V - 5 V.
- Output: Analógica y digital.
- Potenciómetro para ajustar el umbral de sensibilidad.
- Temperaturas: -40 a +85 °C.
- Dimensiones: 35 x 15 x 14mm.

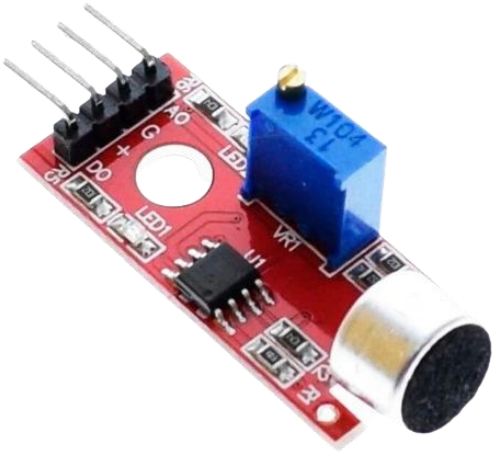


Imagen de un módulo KY-037

ULN2003A

Para el siguiente driver en cuestión, se optó por el mismo ya que permitirá una fácil configuración, así como manejo para nuestro motor a paso, pues el mismo es el puente entre el motor a paso y nuestro ESP32, encargándose así de interpretar las señales pertenecientes al motor para mediante las instrucciones establecidas, permitirnos el giro del mismo

Especificaciones y características

- Cuatro LEDs indicadores de estado.
- Compatible con motor a pasos modelo 28BYJ-48
- Alimentación: 5 Vcc
- Alto: 40 mm
- Ancho: 21 mm
- Espesor: 17 mm

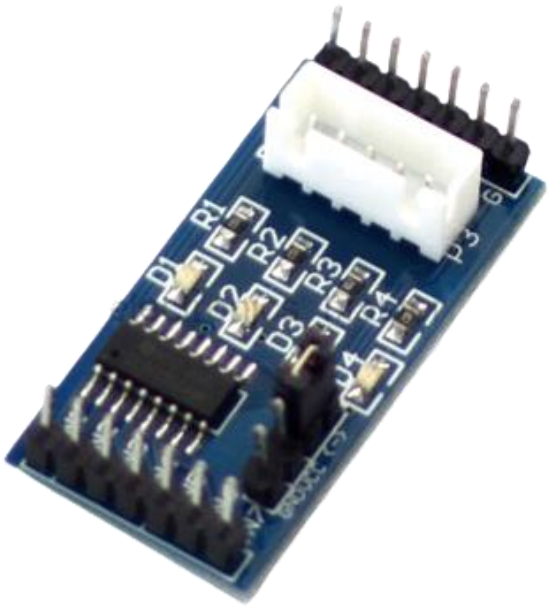


Imagen de driver ULN2003A

Step Motor 28BYJ-48

Motor a paso de tamaño reducido y de cualidad bipolar, sus limitaciones debido al tamaño son evidentes, sin embargo, gracias a su reductor integrado fue capaz de ser útil en este proyecto, el mismo se encuentra encargado del giro que controlara la cámara, con sus 64 pasos por vuelta gracias al reductor antes mencionado permite una precisión justa.

Especificaciones y características

- Ángulo de paso: 5.625°
- Relación de reducción: 1/64
- Diámetro: 28mm
- Alto: 20 mm
- Longitud de eje: 9 mm
- Longitud de cable: 24 cm
- Conector universal de 5 pines
- Alimentación: 5 Vcc

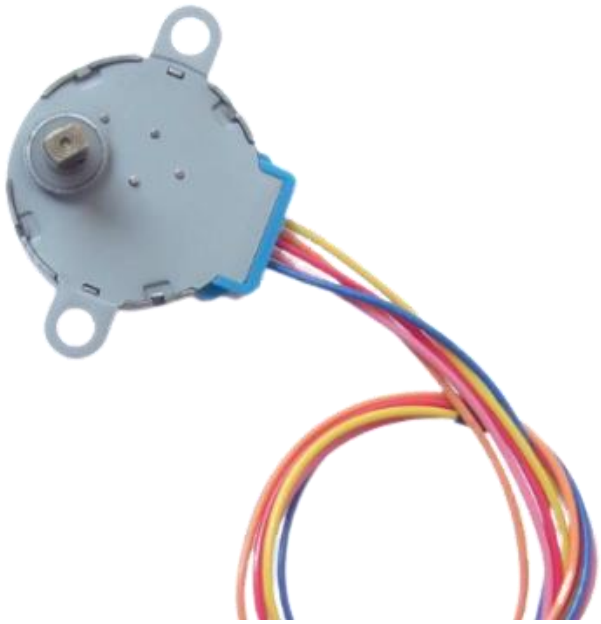


Imagen de motor a pasos modelo 28BYJ-48

Cables DuPont

Los cables son una pieza fundamental en la construcción de cualquier circuito electrónico, por ende, no pudimos prescindir de los mismos, nuestra opción en particular será aquellos con conexión macho hembra.

Imagen de cables DuPont macho-macho

Protoboard

La protoboard permite la construcción fácil y rápida de los circuitos electrónicos, al ser este un prototipo, la protoboard representa una mejor opción que la construcción de una tablilla ya fabricada y manufacturada.

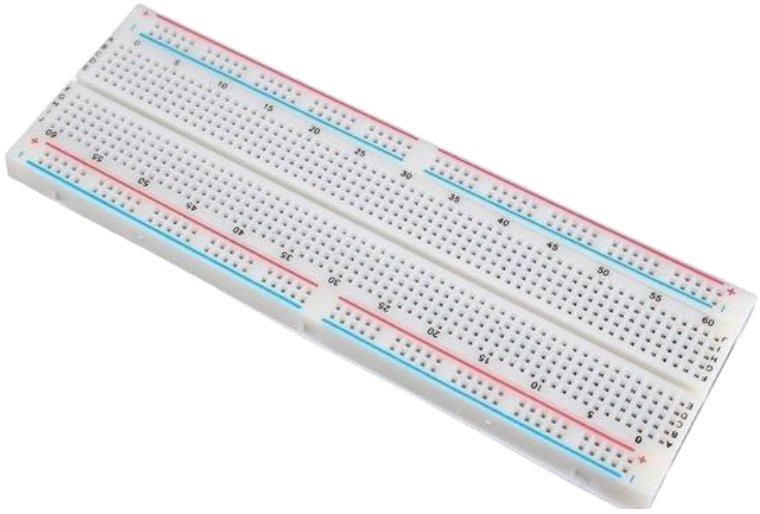


Imagen de protoboard

Fuente de poder 3.3v/5v

Un componente esencial para la implementación de nuestro proyecto, la fuente de alimentación que se muestra en imágenes nos permite suministrar, de manera directa, corriente a nuestra protoboard (5v o 3.3v), permitiéndonos de igual manera alimentar mediante el USB integrado, la alimentación necesaria para nuestro ESP32.

De igual manera esta fuente de alimentación tiene la capacidad para ser alimentada por una batería de 9v, convirtiendo así el circuito a un entorno “wireless”, sin embargo, por motivos de practicidad para el proyecto no se implementó de la manera antes mencionada.

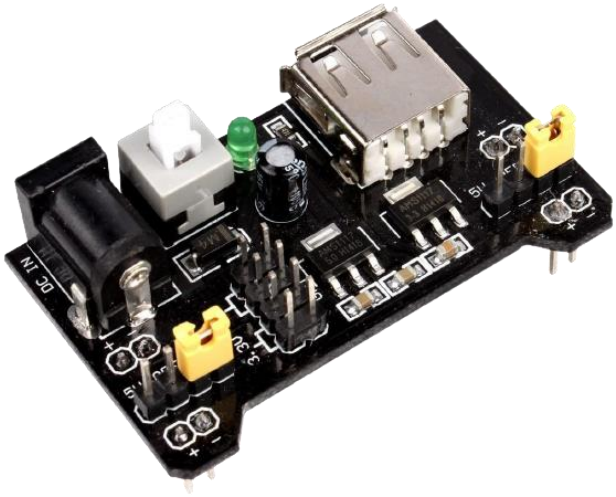


Imagen de fuente de alimentación de 5v – 3.3v

Precio de material

Material	Unidad	Precio total
ESP32	x1	$210.00 MXN
ESP32-CAM	x1	$400.00 MXN
Modulo KY-037	x3	$80.00 MXN
Driver ULN2003A	x1	$45.00 MXN
Step Motor	x1	$115.00 MXN
Cable DuPont	x120	$120.00 MXN
Fuente de poder 3.3v/5v	x1	$115.00 MXN
Protoboard	x1	$60.00 MXN

Descripción de diseño

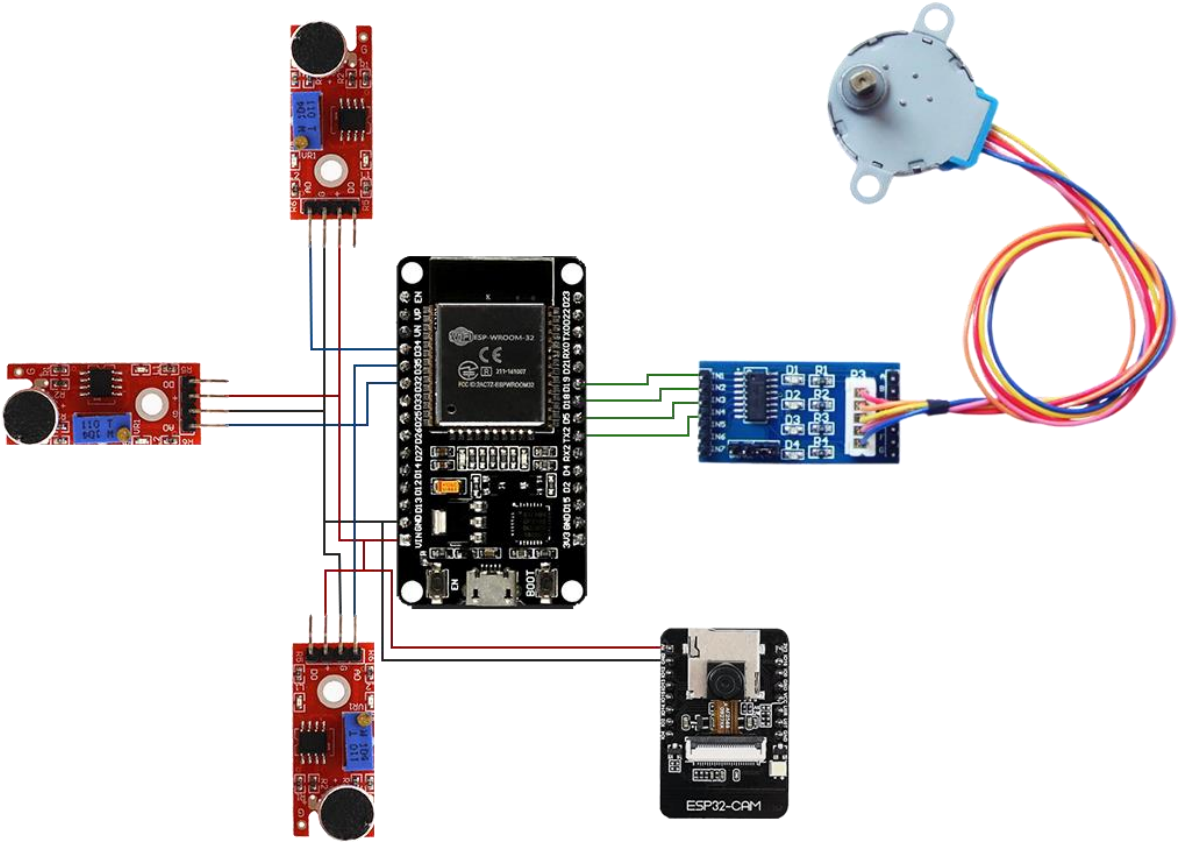
El proyecto consta de 3 micrófonos, conectados a los pines 34,32 y 35 de nuestro ESP32, los cuales corresponden a la posición derecha, izquierda y frente respectivamente, en los cuales leeremos la señal analógica que los mismos proporcionan al captar el sonido (es importante recalcar que para esta implementación se descarta completamente el uso de la señal digital), de igual manera se conectaran a la tierra y 3.3v respectivamente cada uno, ya que estarán captando sonido en todo momento mediante señales analógicas como se mencionó anteriormente, dichas señales serán filtradas automáticamente para solo tomar en cuenta en momentos de picos relevantes para el usuario todo mediante implementación de software, es por eso que no se tiene ningún otro dispositivo de hardware que realice este proceso.

La señal analógica que producirán los micrófonos será de suma importancia, ya que esta se transmitirá a los motores como una instrucción que les indique a que flanco girar, sin embargo, antes de poder proporcionar la señal nuestro motor debe ser capaz de funcionar de manera eficiente, es por eso que se decidió que se implementaría el uso del driver ULN2003A, que permite un control más simple de

nuestro motor a paso, para esto es necesario conectar los pines IN1,2,3 y 4 del driver mencionado anteriormente, en los pines 19,18,5 y 17 de nuestro ESP32, sin olvidar que a nuestro driver también se le tendrá que conectar el conector universal de 5 pines tipo 2510-5Y.

Finalmente se acoplará nuestra ESP32-CAM que se encargará de grabar todo lo que ocurre en nuestro entorno, realizando una transmicion via wifi de manera constante.

Diagrama del circuito



Circuito usado en la realización de "The watcher".

Desarrollo del proyecto

Antes de describir el comparador debemos comprender la transformada de Fourier, ya es necesario emplear esta fórmula matemática para transformar señales entre el dominio del tiempo o espacio y el domino de la frecuencia. El sensor al captar sonido sin filtrar es muy difícil de interpretar, por lo tanto, es muy importante la librería "arduinoFFT" (HTTPS://GITHUB.COM/KOSME/ARDUINOFFT). Básicamente la transformada de Fourier es el espectro de frecuencia de una función. Un buen ejemplo de esto es lo que hace el oído humano, ya que recibe una onda auditiva y la transformó en una descomposición en distintas frecuencias, esto nos ayudara a diferenciar entre sonidos altos – bajos y poder cancelar ciertas frecuencias que nuestros sensores estén detectando.

La transformada de Fourier es una aplicación que hace corresponder a una función *f* con otra función g definida de la manera siguiente:

$$g(\xi) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(x) e^{-i\xi x} dx$$

Entonces lo que nosotros realizamos para mejorar la percepción de distintas frecuencias a medida que pasa el tiempo, la transformada de Fourier contiene todas las frecuencias del tiempo durante el cual existió la señal y las descompone obteniendo el pico más alto o el más bajo y también frecuencias intermedias que fácilmente pueden ser ignoradas.

Para poder construir el comparador secuencial de sonido pasamos por tres diferentes prototipos, mediante la comprobación y análisis de fidelidad de los tres sensores en serie. El tercer prototipo esta optimizado para recibir 1024 bits en muestras, sin embargo, al momento de entrar al comparador y determinar la orientación adecuada para la cámara simplemente tomara 512 muestras mejorando el tiempo de respuesta y cancelar rebotes de voltaje en la bobina del sensor.

A continuación, se describirá a detalle cómo se compone **Comparador secuencial de sonido.**

```
#include <iostream>
#include "arduinoFFT.h"

using namespace std;

///GPIO INPUT
#define right 34
#define left  32
#define front 35
```

Para comenzar incluimos las librerías de C++ y la librería que desarrolla la transformada de Fourier.

En el pin 34 declaramos el uso de la señal analógica del sensor de la derecha, 32 para el sensor de la izquierda y por último 35 para el sensor frontal.

Es muy importante declarar estos pines ya que realizan la conversión analógica-digital que es necesaria para poder pasar esta variable a la función FFT.

```
arduinoFFT FFT = arduinoFFT();

int i ;

const int SAMPLE = 1024;    //
const int tolerance= 2600;  //

//// Real number
double vRight [SAMPLE];
double vLeft  [SAMPLE];     //
double vFront [SAMPLE];

//// Imaginary number
double iRight [SAMPLE];
double iLeft  [SAMPLE];    //
double iFront [SAMPLE];
```

Para terminar con las declaraciones de variables inicializamos la función FFT.

Declaramos las muestras adecuadas en este caso se usaron 1024 bits y la tolerancia es de 2600, con esta tolerancia es posible ignorar frecuencias bajas, como la voz de una persona hasta ruido externo.

En estos arreglos se capturan las entradas analógicas de los sensores, tanto como números reales y números imaginarios que son útiles para la transformada de Fourier.

```
//// Data capture in the Array
void readMicrophones (){
   for (int i = 0 ; i < SAMPLE ; i++){
    vRight[i] = analogRead(right);
    iRight[i] = 0;

    vLeft[i]  = analogRead(left);
    iLeft[i]  = 0 ;

    vFront[i] = analogRead(front);
    iFront[i] = 0 ;
  }
}
```

La función ***readMicrophones*** se encarga de leer los micrófonos e ir introduciendo las cantidades ya manipuladas por el conversor análogo digital que contiene la ESP32, en el periodo de muestreo ya antes declarado.

Esto se hará en cada uno de los 3 micrófonos (KY-037)

Antes de mostrar la funcionabilidad del sistema en el void loop debemos explicar las funciones que contiene la librería “arduinoFFT.h” que transforma nuestras señales analógicas que contienen los arreglos de lo sensores derecha, izquierda y frontal.

Esta función como veremos a continuación se hará la llamada antes del comparador para poder manipular los arreglos ya transformados por Fourier.

```
//.1
void fourierRight(){
  FFT.Windowing           (vRight, SAMPLE, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
  FFT.Compute             (vRight, iRight, SAMPLE, FFT_FORWARD);
  FFT.ComplexToMagnitude  (vRight, iRight, SAMPLE);
}
//.1
void fourierLeft(){
  FFT.Windowing           (vLeft, SAMPLE, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
  FFT.Compute             (vLeft, iLeft, SAMPLE, FFT_FORWARD);
  FFT.ComplexToMagnitude  (vLeft, iLeft, SAMPLE);
}
//.1
void fourierFront(){
  FFT.Windowing           (vFront, SAMPLE, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
  FFT.Compute             (vFront, iFront, SAMPLE, FFT_FORWARD);
  FFT.ComplexToMagnitude  (vFront, iFront, SAMPLE);
}
```

Fragmento de código de “TheWatcher.cpp”.

La secuencia que sigue el programa es muy fácil de entender:

- Hace la llamada de la función ***readMicrophones*** y comienza a leer las entradas de estos sensores.
- Después transforma la frecuencia pura del micrófono derecho con la llamada de la función ***fourierRight***.
- Transforma la frecuencia pura del micrófono izquierdo con la llamada de la función ***fourierLeft***.
- Transforma la frecuencia pura del micrófono izquierdo con la llamada de la función ***fourierFront***.
- En el paso número 5 tenemos un ciclo *for* que comparara las tres señales tomando solamente 512 bits de muestras de los sensores, vemos que esta función compara la señal de los tres sensores que solamente responderán la tolerancia declarada anteriormente y a partir del índice 2 de los arreglos empezará a tomar lectura comparando los tres sensores determinando cual es el que tiene el pico más alto, esta función es el MASTER COMPARATOR que lograra el funcionamiento adecuado para la orientación de la cámara.

```
53  void loop() {
54    //// Reading Signals
55    readMicrophones();
56    ////.1 Fast Fourier Transform Right
57    fourierRight();
58    ////.1 Fast Fourier Transform Left
59    fourierLeft();
60    ////.1 Fast Fourier Transform Front
61    fourierFront();
62
63    //// MASTER COMPARATOR
64    //.3For better performance remove all the couts
65    for (i = 0 ; i < 512 ; i++){
66      if ( vRight [i] > tolerance && i > 1 ){ cout << vRight[i] << " Right" << endl;}
67      if ( vLeft  [i] > tolerance && i > 1 ){ cout << vLeft[i]  << " Left"  << endl;}
68      if ( vFront [i] > tolerance && i > 1 ){ cout << vFront[i] << " Front" << endl;}
69    }
70  }
```

Fragmento de código de "TheWatcher.cpp".

Finalmente para la implementación del uso de motores nos percatamos que teníamos diferentes opciones que nos brindarían resultados “similares” como lo son:

- Motor DC [con uso de reductores de velocidad]
- Servo motor
- Motor a paso

Sin embargo, existieron determinantes que nos hicieron inclinarnos por un motor a paso el cual se ha mencionado en reiteradas ocasiones, más que nada por la precisión, ruido apenas perceptible y velocidad justa que resultaba ideal para el movimiento de la cámara incorporada.

El funcionamiento de nuestro motor es muy simple ya principalmente se encuentra controlado por las siguientes funciones:

```
void mover_Norte() {
  cout << "Moviendo a Norte" << endl; // imprime avis
ando que se moverá hacia enfrente
  if (pos != 0){ // compara la posición para saber si
   no esta enfrente ya
    switch(pos){ // dependiendo la posición se va al
    caso que sea necesario
      case 1: // si está a la derecha es 1
        myStepper.step(512); // lo mueve un cuarto de
        vuelta a la izquierda
      break;
      case -1: // si está a la izquierda es-1
        myStepper.step(512); // lo mueve un cuarto de
        vuelta hacia la derecha
      break;
    }
    pos = 0; // se ajusta la posición para marcar que
    esta al norte
  }
```

Fragmento de código de “TheWatcher.cpp”.

La cual a groso modo se encarga de evaluar la posición comparando donde es que se encuentra captando nuestra señar analógica y determinando si el movimiento será hacia el flanco derecho o izquierdo dependiendo esto del micrófono que este proporcionando la señal.

Este fragmento de código junto a él que se mostrara a continuación serán los encargados de controlar en su totalidad el movimiento de nuestro motor.

```
  for (i = 0 ; i < 512 ; i++){
    if ( vRight [i] > tolerance && i > 1 ){
      mover_Oeste();
      cout << vRight[i] << " Right" << endl;
    }
    if ( vLeft  [i] > tolerance && i > 1 ){
      mover_Este();
      cout << vLeft[i]  << " Left"  << endl;
    }
    if ( vFront [i] > tolerance && i > 1 ){
      mover_Norte();
      cout << vFront[i] << " Front" << endl;
    }
  }
}
```

Fragmento de código de "TheWatcher.cpp".

Código GitHub

En la siguiente liga podremos encontrar el repositorio que contiene todos los códigos pertenecientes al proyecto, así como el que se muestra en cuestión.

HTTPS://GITHUB.COM/BSALD205/THE-WATCHER

Planeación

La planeación estratégica que impartimos para este proyecto estuvo por 5 pasos específicos que utilizamos al hacer los prototipos:

1. Define los requerimientos y variables.
2. Define las herramientas para el diseño y testeo.
3. Diseña el prototipo de tu idea.
4. Testea el prototipo.
5. Analiza los resultados y aprendizajes

Mediante la implementación de estos pasos fuimos capaces de avanzar de manera más eficiente, enfocándonos un prototipo a la vez, utilizando el aprendizaje aprendido para la toma de decisiones que nos llevarían a el siguiente prototipo.

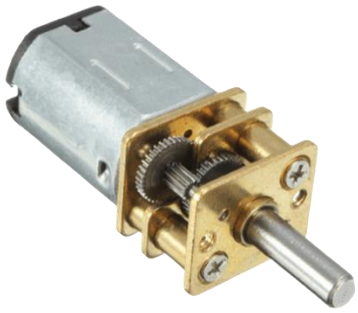
Si bien no todo salió bien al primer intento logramos culminar satisfechos con el prototipo final

Problemas

Al realizar este proyecto los problemas no faltaron, como en cada desarrollo siempre se presentan imprevistos que, si no se llegan a tomar de la manera correcta, estos pueden representar graves y serios problemas para los desarrolladores, en nuestro caso

Como ya se mencionó con anterioridad se presentaron diferencias sustanciales en cuestiones de motores, pues ya que las 3 opciones que se van a denotar son buenas, contemplan usos para ámbitos diferentes debido a sus características.

- Motor DC [con uso de reductores de velocidad].

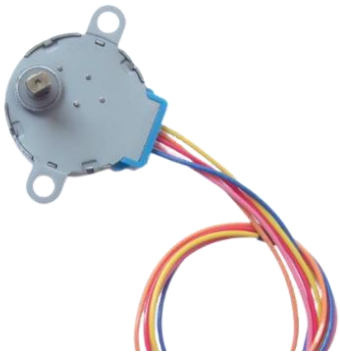


El presente motor es una gran opción para funciones por ejemplo de giro de ruedas en un carro a control remoto, debido a la alta velocidad de giro y poco control de precisión, nosotros tuvimos que optar por el uso de reductores de giro que permitiera un uso acorde al proyecto, sin embargo, aun a pesar del uso del reductor el uso forzoso de las variables arraigadas con el tiempo en cuestiones de giro resultaron en una imprecisión, que resultaría desfavorable para el giro de la cámara.

- Servo motor.

Sin duda alguna a primera vista el uso de servo motores pareceria una opcion apta para el esencario planteado, sin embargo, la inestabilidad que presenta al hacer el giro (en ocacion presentaba un rebote al llegar a la posicion designada) que si bien podria ser corregida con programacion avanzada fue solo un indicio de que no podria ser apto para este entorno, pues finalmente este aunado al fuerte sonido producido con si giro que ocasionaba interferencias con nuestro microfono, termino por descartar esta opcion, aun a pesar de sus grandes prestaciones en cuestiones de facilidad a la hora del “contorl” y por su puesto su reducido tamaño capaz de tener gran torque.

- Motor a paso



Aun a pesar de que fue la opción que tomamos finalmente, nos es perfecto, ya que su uso es bastante irregular en ocasiones, pues al funcionar con pasos en ocasiones se descalibra totalmente y no proporciona el movimiento que se busca.

- Conexiones

Sin dudarlo uno de los problemas más grandes a los cual nos enfrentamos fue debido a que las conexiones no estaban listas para someterse a constantes traslados, es decir que al no estar ni soldados ni sujetos a nada más que los dispositivos que se tenían, se presentaron múltiples problemas de conectividad, sin embargo creemos seriamente que de continuar con más prototipos, en algún momento podríamos ser capaces de remplazar estas conexiones improvisadas por unas que permitan la operabilidad correcta de nuestro dispositivo en cuestión.

Resultados y conclusiones

En conclusión, sin duda alguna este proyecto represento un reto para el equipo, pues comprendimos el complejo mundo de todos los dispositivos de IoT que nos rodean, así como el arduo trabajo en materia de programación que viene aunado a cada uno de ellos, sin embargo, hemos de decir que disfrutamos de manera encarecida el poder desarrollar algo que en verdad pudiese llegar a tener una implementación en el día a día de alguna persona ahí afuera, es por eso que pretendemos continuar en alguna otra ocasión implementando mejoras para nuestros prototipos, tal vez en algún momento donde no tengamos una fecha límite que nos permita trabajar de manera más relajada.

Claramente no todo es bueno como todo en la vida, pues fueron muchos los retos que se presentaron a la hora de realizar el proyecto, pues fuimos espectadores de una serie de eventos desafortunados con fallas tras fallas, en diferentes dispositivos y diferentes aspectos de proyecto, desde la programación con múltiples intentos, hasta las conexiones electrónicas de los dispositivos.

Es por eso que reconocemos la gran experiencia que se adquirido en este proyecto, que pretendemos sea la entrada a tal vez una relación laboral en el futuro, pues como ya se mencionó anteriormente, en este momento tenemos más experiencia en el tema de lo que teníamos en un momento ya que esto no es nada más y nada menos que los cimientos del comienzo de una vida por delante, trabajando en más dispositivos, conociendo más lenguajes de programación para tal vez algún día poder implementar todo lo que hoy aprendimos en alguna empresa que sea capaz de valorar nuestro tiempo y conocimiento.

Video demostrativo:

https://drive.google.com/drive/folders/1mUfjZ1qXcshaJfJZpk8EBfbX0C4k85_U?usp=sharing

Referencias

https://freed.tools/blogs/ux-cx/prototipo#:~:text=cumple%20o%20no.-,Metodolog%C3%ADa%20de%20prototipo%20o%20prototipado,medir%20y%20ajustar%20un%20plan.

https://www.digikey.com.mx/es/articles/how-to-select-and-use-the-right-esp32-wi-fi-bluetooth-module

https://programarfacil.com/esp32/esp32-cam/#:~:text=ESP32%2DCAM%2C%20es%20un%20dispositivo,podremos%20almacenar%20fotos%20o%20videos.

https://www.tme.eu/es/news/library-articles/page/41861/Motor-paso-a-paso-tipos-y-ejemplos-del-uso-de-motores-paso-a-paso/