Proyecto final

Juan David Medina - 7004022
est.juan.medina@unimilitar.edu.co
Andrés Felipe Bernal - 7003748
est.andres.bernal1@unimilitar.edu.co

Resumen: En el siguiente proyecto se podrá observar la creación desde cero de un automóvil eléctrico, en el que su movimiento y funcionamiento se centrará en la programación realizada en el programa de Keil Uvision, con ayuda de la tarjeta programable STM32F767, tendremos un acercamiento mucho mejor hacia las diferentes herramientas del proyecto como los GPIO, ADC, SPI, TIMERS, I2C, DAC, EXTI y USART, con la combinación de dichas herramientas para el funcionamiento completo del automóvil y sus componentes que se hablaran mas tarde, al ser demasiados con diferentes funciones.

Palabras clave: Stm32F767 – Keil Uvision – leds – motor – corriente – movimiento.

Abstract: In the following project, the creation of an electric car from scratch will be observed, in which its movement and operation will be centered on the programming carried out in the Keil Uvision program, with the help of the programmable STM32F767 board. We will have a much better approach to the different tools of the project such as GPIO, ADC, SPI, TIMERS, I2C, DAC, EXTI, and USART, with the combination of these tools for the complete operation of the car and its components, which will be discussed later, as there are too many with different functions.

Keywords: Stm32F767 – Keil Uvision – leds – motor– current – movement.

I. INTRODUCCION.

En el siguiente trabajo veremos el uso de diferentes herramientas para la programación del proyecto y de las diferentes funciones a cumplir con el automóvil, se realizarán los registros para la configuración de ayudas como los GPIO, el ADC y DAC, el SPI, los TIMERS, el I2C, las EXTI y USART, cada función nos da un amplio uso de los componentes a manejar.

El funcionamiento básicamente se basa en motores DC para el movimiento hacia delante o hacia atrás, servomotores para la dirección derecha e izquierda y para el mástil que agarrara objetos, el sensor ultrasónico para tener a una distancia prudente el objeto a agarrar, la fotorresistencia en caso de poca iluminación activar un led que guiara el carro y por ultimo leds que indicaran la dirección a la que gira el carro y su stop, por último se trabajo el uso de Python como herramienta de creación de interfaz gráfica para emplear el controlador del automóvil, en si en estos componentes y funciones se reduce el proyecto realizado, el lenguaje de STM32F7 es muy amplio y nos brinda un nuevo mundo como lo siguiente a observar.

II. MARCO TEORICO.

STM32:

El STM32 es una familia de microcontroladores de 32 bits desarrollados por la compañía STMicroelectronics. Estos microcontroladores ofrecen una amplia variedad de características y opciones de conectividad para una amplia gama de aplicaciones. Son muy populares en el campo de la electrónica, la robótica y la industria, gracias a su potencia y versatilidad.

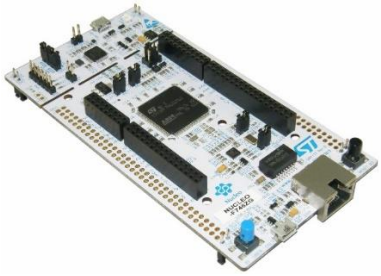

Imagen 1. STM32

KEIL:
Keil uVision es un entorno integrado de desarrollo (IDE) utilizado principalmente para programar microcontroladores de la familia ARM, aunque también admite otros microcontroladores. Fue desarrollado por la compañía Keil, que fue adquirida por ARM en 2005 y ahora es parte de la empresa de software de desarrollo ARM. Keil uVision ofrece una plataforma completa para el desarrollo de proyectos de microcontroladores, que

incluye herramientas de compilación, depuración, simulación y programación. Es una herramienta popular entre los desarrolladores de sistemas embebidos debido a su facilidad de uso y su amplio soporte para microcontroladores de diferentes fabricantes**.**

RESISTENCIAS:
Una resistencia es un componente eléctrico pasivo diseñado para limitar el flujo de corriente eléctrica en un circuito. Su función principal es ofrecer resistencia al paso de la corriente, lo que regula la cantidad de corriente que

fluye a través de ella y también afecta la caída de voltaje en el circuito.

Imagen 2. Resistencias.

LEDS:
Un LED, o Diodo Emisor de Luz, es un dispositivo semiconductor que emite luz cuando pasa corriente eléctrica a través de él. Su capacidad para generar luz en diversos colores, su eficiencia energética y durabilidad lo convierten en una opción ampliamente utilizada en indicadores visuales, pantallas, iluminación y una variedad de aplicaciones electrónicas. Su tamaño compacto, encendido instantáneo y la posibilidad de controlar su intensidad hacen que los LEDs sean esenciales en la tecnología moderna.

Imagen 3. Leds

MOTOR DC:
Un motor de corriente continua (DC) es un dispositivo electromecánico que convierte la energía eléctrica en energía mecánica mediante el uso de corriente continua. Este tipo de motor se utiliza en una amplia variedad de aplicaciones, desde juguetes hasta vehículos eléctricos.

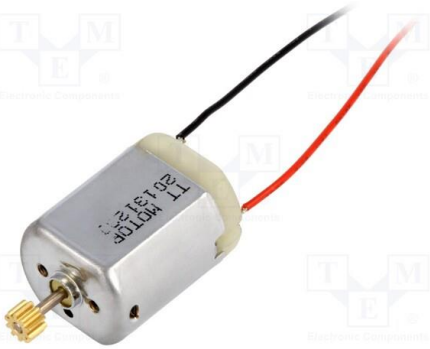

Imagen 4. Motor DC.

SERVOMOTOR:
Un servomotor es un dispositivo de control que utiliza retroalimentación para controlar la posición, velocidad y/o aceleración de un mecanismo. Estos motores se utilizan en una amplia variedad de aplicaciones, como en la industria de la robótica, sistemas de control de vuelo de aeronaves, sistemas de dirección de automóviles, entre otros.

Imagen 5. Servomotor.

SENSOR ULTRASÓNICO:

Un sensor ultrasónico es un dispositivo que emite ondas de sonido de alta frecuencia y utiliza el eco de estas ondas para medir distancias, detectar objetos y realizar otras funciones de detección. Estos sensores se utilizan en una amplia gama de aplicaciones, como en sistemas de estacionamiento automático, sistemas de alarma, sistemas de navegación de robots, entre otros.

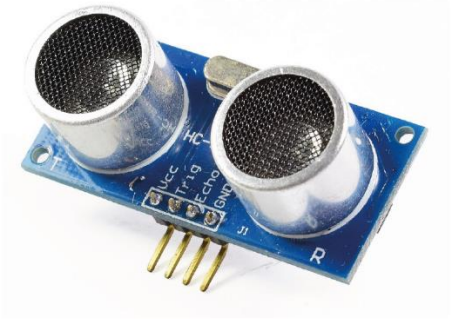

Imagen 6. Sensor ultrasónico.

FOTORRESISTENCIA:

Una fotorresistencia, también conocida como sensor de luz o LDR (del inglés Light Dependent Resistor), es un componente electrónico que varía su resistencia eléctrica en función de la intensidad de la luz que incide sobre él. Se utiliza en circuitos de control automático de iluminación, sistemas de detección de luz, entre otros.

Imagen 7. Fotorresistencia.

III. PROCEDIMIENTO.

Para el desarrollo del proyecto, se comenzó por definir los pines a usar, quedando como resultado la siguiente tabla:

PA	Parametros	Usos	STM32	Ubicacion	PIN	MODER	Alternante
PB	ADC 1		ADC1	CN11 - 30	PB1	11	-
PC	ADC 2	Distancia	ADC2	CN7 - 14	PA7	11	-
PD	ADC 3	Fotoresistencia	ADC3	CN7 - 1	PA3	11	-
	DAC 1	Iluminacion	DAC	CN7 - 17	PA4	11	-
	USART	PC Interfaz	USART3 RX	CN12 - 10	PD8	10	7
	USART	PC Interfaz	USART3 TX	CN11 - 69	PD9	10	7
	TIMMER 1	PWM servo dir	TIM3	CN7 - 12	PA6	10	2
	TIMMER 2	PWM servo g1	TIM5	CN10 - 29	PA0	10	2
	TIMMER 3	PWM servo g2	TIM4	CN10 - 13	PB6	10	2
	GPIO 1	Enable H1	En1	CN11 - 38	PC0	1	-
	GPIO 2	Enable H2	En2	CN11 - 36	PC1	1	-
	GPIO 3	IN H1	I1	CN11 - 35	PC2	1	-
	GPIO 4	IN H2	I2	CN11 - 37	PC3	1	-
	GPIO 5	IN H3	I3	CN12 - 34	PC4	1	-
	GPIO 6	IN H4	I4	CN12 - 6	PC5	1	-
	GPIO 7	LED 1	Left	CN12 - 4	PC6	1	-
	GPIO 8	LED 2	Rigth	CN12 - 19	PC7	1	-
	GPIO 9	LED 3	Back	CN12 - 2	PC8	1	-

Tabla 1. Asignación de pines necesarios

Una vez se tenia esta tabla se podía realizar la función necesaria para habilitar los puertos y asignar el registro MODER, OSPEEDR y AFR[x] según fuera necesario:

```
77  void Config_GPIO(){
78      RCC->AHB1ENR |= 0x2F;
79
80      GPIOA->MODER |= 0xE3C2;
81      GPIOA->OSPEEDR |= 0xAAAAAAAA;
82      GPIOA->AFR[0] |= 0x22222222;
83
84      GPIOB->MODER |= 0x200F;
85      GPIOB->OSPEEDR |= 0xAAAAAAAA;
86      GPIOB->AFR[0] |= 0x22222222;
87
88      GPIOC->MODER |= 0x15555;
89      GPIOC->OSPEEDR |= 0xAAAAAAAA;
90
91      GPIOD->MODER |= 0xA0000;
92      GPIOD->OSPEEDR |= 0xAAAAAAAA;
93      GPIOD->AFR[1] |= 0x77;
94  }
```

Sección 1. Función encargada de habilitar y configurar los pines

De esta forma, se procedió a crear diferentes funciones que permitían configurar los registros necesarios de cada uno de los parámetros requeridos en el proyecto, los cuales fueron ADC, Timer, DAC y USART:

```
void Config_ADC(){
  RCC->APB2ENR|= 0x700;// ADC1...2...3 habilitados
  ADC->CCR |= (0<<16); // PCLK2 dividido por 2

  ADC1->CR1 |= (0<<24); // 12 bits de resolucion
  ADC2->CR1 |= (0<<24);
  ADC3->CR1 |= (0<<24);

  ADC1->CR2 |= (1<<0)|(1<<1)|(1<<5)|(0<<11);  // ADC encendido,
  ADC2->CR2 |= (1<<0)|(1<<1)|(1<<5)|(0<<11);  //conversion continua,
  ADC3->CR2 |= (1<<0)|(1<<1)|(1<<5)|(0<<11);  //EOCS habilitado,
                                              //alineacion a la derecha.

  ADC1->SMPR2 |= (7<<27); // ADC1 IN9 con 480 ciclos
  ADC2->SMPR2 |= (7<<12); // ADC2 IN4 con 480 ciclos
  ADC3->SMPR2 |= (7<<9); // ADC3 IN3 con 480 ciclos

  ADC1->SQR1 |= (0<<0); // L=0, numero de conversiones = 1
  ADC1->SQR2 |= (0<<0); // Nada
  ADC1->SQR3 |= (1<<0); // ADC1 IN9 como primera conversion

  ADC2->SQR1 |= (0<<0); // L=0, numero de conversiones = 1
  ADC2->SQR2 |= (0<<0); // Nada
  ADC2->SQR3 |= (7<<0); // ADC2 IN4 como primera conversion

  ADC3->SQR1 |= (0<<0); // L=0, numero de conversiones = 1
  ADC3->SQR2 |= (0<<0); // Nada
  ADC3->SQR3 |= (3<<0); // ADC3 IN3 como primera conversion
}
```

Sección 2. Función encargada de configurar el ADC1, ADC2 y ADC3

```
void Config_Timer3(){//PA6    Garra 1
  RCC->APB1ENR |=(0x1<<1);

  TIM3->ARR = 4999;
  TIM3->PSC = 15;
  TIM3->EGR = 0x1;
  TIM3->CCMR1 = 0x6060;
  TIM3->CCER = 0x1111;
  TIM3->CR1 = 0x1;
}
void Config_Timer4(){//PB6    Garra 2
  RCC->APB1ENR |= (0x1<<2);

  TIM4->ARR = 20000 - 1;
  TIM4->PSC = 16 - 1;
  TIM4->EGR = 0x1;
  TIM4->CCMR1 = 0x6060;
  TIM4->CCMR2 = 0x6060;
  TIM4->CCER = 0x1111;
  TIM4->CR1 = 0x1;
}
```

Sección 3 y 4. Funciones encargadas de la configuración del Timer3 y Timer4.

```
void Config_Timer5(){ //PA0  Direccion
  RCC->APB1ENR |= (0x1<<3);

  TIM5->ARR = 20000- 1;
  TIM5->PSC = 16 - 1;
  TIM5->EGR = 0x1;
  TIM5->CCMR1 = 0x6060;
  TIM5->CCMR2  = 0x6060;
  TIM5->CCER = 0x1111;
  TIM5->CR1 = 0x1;
}
void Config_DAC(){
  RCC->APB1ENR |= (0x1<<29);// DAC Enable
  DAC->CR |= 0x1;
}
```

Sección 5 y 6. Funciones encargadas de la configuración del Timer5 y DAC

```
void Config_USART3(){
  RCC->APB1ENR |= (0x1<<18);
  USART3->CR2 &= ~(0x1<<12);
  USART3->CR2 &= ~(0x1<<13);
  BAUDRATE = (16000000/9600)+1;
  USART3->BRR |= BAUDRATE;
  USART3->CR1 |= 0xD;           // Transmision, recepcion, stop mode, enable USART
  USART3->CR1 |= (0x1<<5);      // Interrupcion (recepcion)
  USART3->CR1 &= ~(0x1<<15);  // Over8 = 0

  NVIC_SetPriority(USART3_IRQn,2);
  NVIC_EnableIRQ(USART3_IRQn);
}
extern "C"{
  void USART3_IRQHandler(void){
      if(USART3->CR1 == 0x2D){
        while (USART3->ISR & USART_ISR_RXNE){
          in =(USART3->RDR & 0xFF);
        }
      }
  }
}
```

Sección 7 y 8. Funciones encargad de la configuración del USART3 y la interrupción asociada para recepción de información.

Después de comprobar el correcto funcionamiento de estas funciones, se procedió a generar el programa principal por el cual el vehículo fue controlado mediante una interfaz creada en Python. El control permite avanzar, retroceder, girar a la derecha e izquierda, además de un botón de secuencia por el cual se indica al vehículo que debe accionar los servomotores para recoger el objetivo. Las acciones de desplazamiento se determinaban con el sentido de giro de los motores conectados a un puente H y la dirección se indicaba mediante un servomotor.

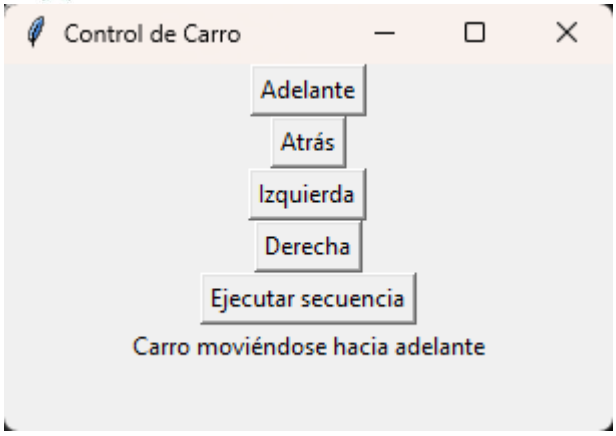

Figura 1. Interfaz encargada de controlar dirección y secuencia del carro

Estas son las acciones controladas por el usuario, sin embargo, se usaron los sensores antes indicados para encender un led variando su intensidad según que tan oscuro este el ambiente, y el sensor de distancia impide que el carro impacte directamente contra los objetivos que estén demasiado cerca.

IV. RESULTADOS.

Una vez se completó el código y el montaje necesario para su funcionamiento, estos fueron los resultados:

El envío de información para el control de carro mediante la interfaz era correcto, por lo que la información que llegaba a la STM32 era suficiente para

iniciar el funcionamiento del carro, el puente H tenía una respuesta adecuada, aunque por problemas con la alimentación tenían problemas los motores para romper la inercia y retroceder, aunque para los demás modos de control los motores podían girar lo suficiente para mover el vehículo en retroceso podía quedarse estático. Las luces indicadoras en el vehículo también funcionaban correctamente, encendiendo o apagando según el modo del vehículo.

Para el control de la secuencia se tenía la configuración correcta de los Timer4 y Timer5, sin embargo, nuevamente por temas de alimentación de energía, no fue posible conectar todos los servomotores, por lo que en la presentación final se decidió no usar el brazo encargado de ejecutar la secuencia.

Otro de los inconvenientes se presentó en los ADC, ya que al tener destinados tres ADC, la configuración de cada uno de ellos era correcta y estaban en capacidad de leer información simultáneamente, sin embargo, solo contábamos con la fotorresistencia, ya que el sensor ultrasónico requería de una configuración adicional por lo que este apartado del vehículo tampoco fue completamente implementado.

Imagen 8. Primera Maqueta

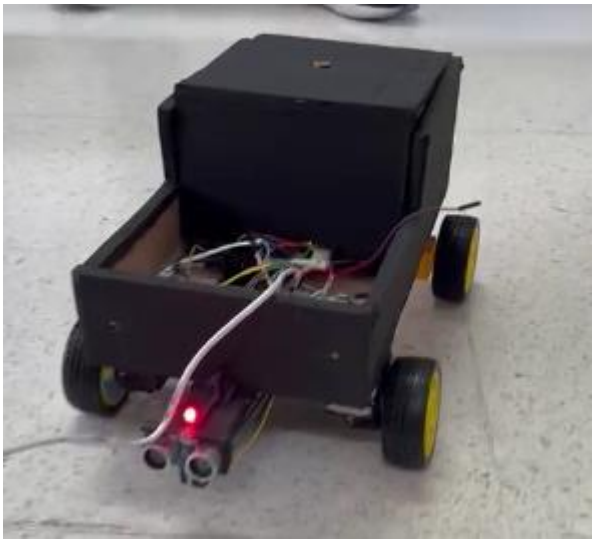

Imagen 9. Indicador de retroceso

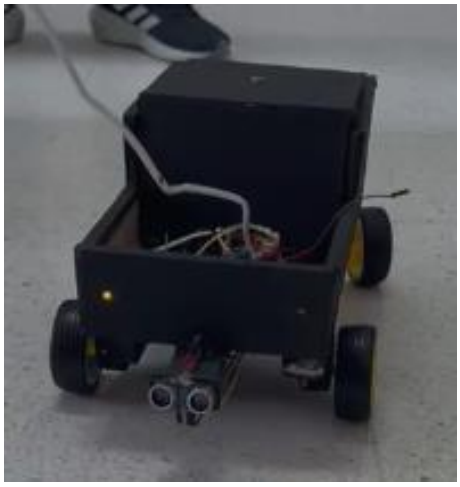

Imagen 10. Giro a la Izquierda e indicador

Imagen 11. Giro a la Derecha e indicador

V. CONCLUCIONES.

Después de desarrollar este proyecto, se llegó a las siguientes conclusiones:

ADC (Convertidor Analógico-Digital):

- Permite la conversión de señales analógicas en digitales, esencial para leer datos de sensores como temperatura, luz, presión, etc.
- Ofrece alta precisión y resolución en la conversión, permitiendo la captura precisa de datos del entorno.
- El STM32 tiene múltiples canales ADC, lo que posibilita la lectura simultánea de múltiples señales analógicas.

DAC (Convertidor Digital-Analógico):

- Transforma señales digitales en señales analógicas, útiles para la generación de voltajes o señales que controlan actuadores como motores o sistemas de audio.
- Permite una salida controlada y precisa de señales analógicas desde el microcontrolador.

USART (Universal Synchronous/Asynchronous Receiver/Transmitter):

- Proporciona capacidades de comunicación serial, tanto síncrona como asíncrona, permitiendo la conexión con otros dispositivos periféricos o módulos.
- Es útil para la comunicación con otros microcontroladores, sensores, módulos GPS, dispositivos Bluetooth, entre otros.

GPIO (General Purpose Input/Output):

- Ofrece pines configurables que pueden funcionar como entradas o salidas digitales.
- Esencial para interactuar con el entorno exterior al leer sensores o controlar dispositivos como LED, relés, botones, etc.

Alimentación:

- Dependiendo de los componentes usados, es necesario determinar una fuente de alimentación externa, así como definir si es necesario tener una alimentación externa para la STM32, que permita el cierre completo de la estructura accediendo a las STM32 desde medios inalámbricos.

VI. BIBLIOGRAFÍA

1. crodriguez, "¿Qué es un motor DC y para qué sirve?," *SDI*, Aug. 25, 2022. https://sdindustrial.com.mx/blog/que-es-un motor-dc-y-para-que-sirve/.
2. "Servomotores: Héroes Silenciosos de la Tecnología Moderna," *aula21 | Formación para la Industria*, Jun. 08, 2023. https://www.cursosaula21.com/que-es-un-servomotor/.
3. "¿Qué es un sensor ultrasónico? | Fundamentos del sensor: Guía de sensores para fábricas clasificados por principios KEYENCE," *Keyence.com.mx*, 2023. https://www.keyence.com.mx/ss/products/sensor/sensorbasics/ultrasonic/info/.
4. Ingeniería Mecafenix, "Que es una fotorresistencia o LDR - Ingeniería Mecafenix," *Ingeniería Mecafenix*, May 27, 2022.https://www.ingmecafenix.com/electronica/componentes/fotoresistencia/.
5. "STM32F7x7STMicroelectronics," *STMicroelectronics*, 2023. https://www.st.com/en/microcontrollers-microprocessors/stm32f7x7.html.
6. "Documentation – Arm Developer," *Arm.com*, 2023.https://developer.arm.com/documentation/11407/0538/User-Interface/uVision-GUI.