UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROJETO FINAL
BRAÇO ROBÔTICO OPERANDO EM LINHA DE DISTRIBUIÇÃO

Alexandre Basílio da Silva Júnior
Melquisedeque Leite e Silva

19 de dezembro de 2024

Projeto Final

Braço Robôtico Operando em Linha de Distribuição

Este documento descreve um projeto final do módulo inicial da capacitação em sistemas embarcados oferecido pelo Virtus que utilizou um microcontrolador STM32 para controlar periféricos, que englobam os assuntos ministrados durante as aulas da capacitação. O objetivo é explorar o uso dos periféricos e das tecnicas aprendidas durante os módulos, com o intuito de demonstrar uma aplicação prática na engenharia e simular um possível cenário real.

Autores:
Alexandre Basílio da Silva Júnior
Melquisedeque Leite e Silva

Professores:
Rafael Bezerra Correia Lima
Moacy Pereira da Silva
Alexandre Sales Vasconcelos

Conteúdo

1 Introdução

Este relatório apresenta o desenvolvimento de um projeto prático utilizando uma garra robótica controlada por uma placa STM32, com o objetivo de detectar, separar e classificar objetos conforme seu tipo. A implementação foi realizada utilizando a biblioteca HAL da STM32, aproveitando o periférico de temporização (TIM) para acionar a garra de forma precisa. O sistema de detecção de objetos foi desenvolvido com base em sensores, como sensores ultrassônicos ou de proximidade, que ativam a garra assim que um objeto é identificado. Após o acionamento, a garra coleta o objeto e o classifica com base nas características do produto, como forma, tamanho ou material.

O trabalho começou com a configuração dos periféricos de entrada e saída da STM32, incluindo os sensores de detecção e o controle do motor responsável pela movimentação da garra. Em seguida, foram implementadas rotinas para tratar os sinais de entrada dos sensores e acionar o motor de maneira eficiente. A separação dos objetos foi realizada por meio de um algoritmo que analisa as características dos produtos detectados.

2 Desenvolvimento e Metodologia

2.1 Fundamentação Teórica

2.1.1 Timer

O temporizador, comumente referido como *timer*, é um periférico essencial em sistemas embarcados, utilizado para medir intervalos de tempo, gerar eventos periódicos e controlar sinais de saída com precisão. Em sua essência, um *timer* é composto por um contador que incrementa ou decrementa seu valor em função de um sinal de relógio (*clock*), permitindo o controle temporal de diferentes processos no sistema.

2.1.2 Funcionamento Básico

Um *timer* opera com base em um sinal de *clock* de frequência f_{clock}, que determina a taxa de incremento ou decremento do contador. A frequência do *timer* pode ser ajustada utilizando um *prescaler*, que reduz a frequência do sinal de *clock* para alcançar uma resolução de tempo adequada. O valor do *prescaler* (P) define a relação entre a frequência de entrada e a frequência efetiva do *timer*:

$$f_{\text{timer}} = \frac{f_{\text{clock}}}{P}$$

O contador avança (ou retrocede) em cada ciclo de *clock* do *timer*, e seu valor é comparado com um valor de referência, chamado de *auto-reload register* (ARR). Quando o contador atinge o valor do ARR, ele pode gerar um evento, como a atualização de um sinal de saída ou a ativação de uma interrupção.

Modos de Operação

Os *timers* podem operar em diferentes modos, dependendo da aplicação:

- **Modo Contador:** mede o número de pulsos de um sinal de entrada.
- **Modo Temporizador:** gera eventos em intervalos de tempo fixos.
- **Modo PWM:** utilizado para controlar a largura de pulso em aplicações de modulação por largura de pulso.
- **Modo de Captura/Comparação:** captura eventos externos ou gera saídas com temporização precisa.

2.1.3 PWM

A modulação por largura de pulso (PWM, do inglês *Pulse Width Modulation*) é uma técnica amplamente utilizada em sistemas eletrônicos e de controle para gerar sinais analógicos a partir de dispositivos digitais. O conceito central do PWM baseia-se na variação do **duty cycle**, que é a porcentagem de tempo em que um sinal permanece ativo durante um período completo de operação.

Considere uma onda periódica com período T, onde o nível alto é representado por 1 (estado ativo) e o nível baixo por 0 (estado inativo). Durante o período T, o tempo ativo do sinal é denotado por T_{on}, enquanto o tempo inativo é representado por T_{off}. O período total da onda pode ser descrito como:

$$T = T_{\text{on}} + T_{\text{off}}$$

O **duty cycle** (D) é definido como a razão entre T_{on} e o período total T, podendo ser expressa em porcentagem:

$$D = \frac{T_{\text{on}}}{T} \times 100\%$$

Variando o *duty cycle*, é possível controlar a média de potência entregue a uma carga. Por exemplo:

- Um duty cycle de 50% significa que o sinal está ativo por metade do tempo, resultando em uma potência média reduzida pela metade.
- Um duty cycle de 100% indica que o sinal permanece constantemente ativo, enquanto 0% significa que ele está sempre inativo.

2.1.4 Duty Cycle

O *duty cycle* é um conceito fundamental na modulação por largura de pulso (PWM). Em tradução livre, *duty cycle* significa ciclo de trabalho e, nesse contexto, refere-se à porcentagem de tempo em que um sinal permanece ativo durante um período completo.

Considere uma onda quadrada com período T, onde o nível alto é representado por 1 (sinal ativo) e o nível baixo por 0 (sinal inativo). O período T pode ser decomposto em duas partes: o tempo em que o sinal está ativo (T_{on}) e o tempo em que está inativo (T_{off}). Portanto, podemos expressar o período total da onda da seguinte forma:

$$T = T_{\text{on}} + T_{\text{off}} \tag{1}$$

O *duty cycle* é definido pela razão entre o tempo em que o sinal está ativo e o período total, sendo representado matematicamente pela fórmula:

$$DC = \frac{T_{\text{on}}}{T} \tag{2}$$

Além disso, o tempo em que o sinal permanece inativo (T_{off}) pode ser expresso em função do *duty cycle* pela equação:

$$T_{\text{off}} = T(1 - DC) \tag{3}$$

Essa relação é útil para calcular o tempo em que o sinal permanece em estado baixo com base no período total e na razão do ciclo ativo.

A seguir, é apresentada uma imagem de um programa em Python que ilustra o pulso PWM para diferentes valores de duty cycle.

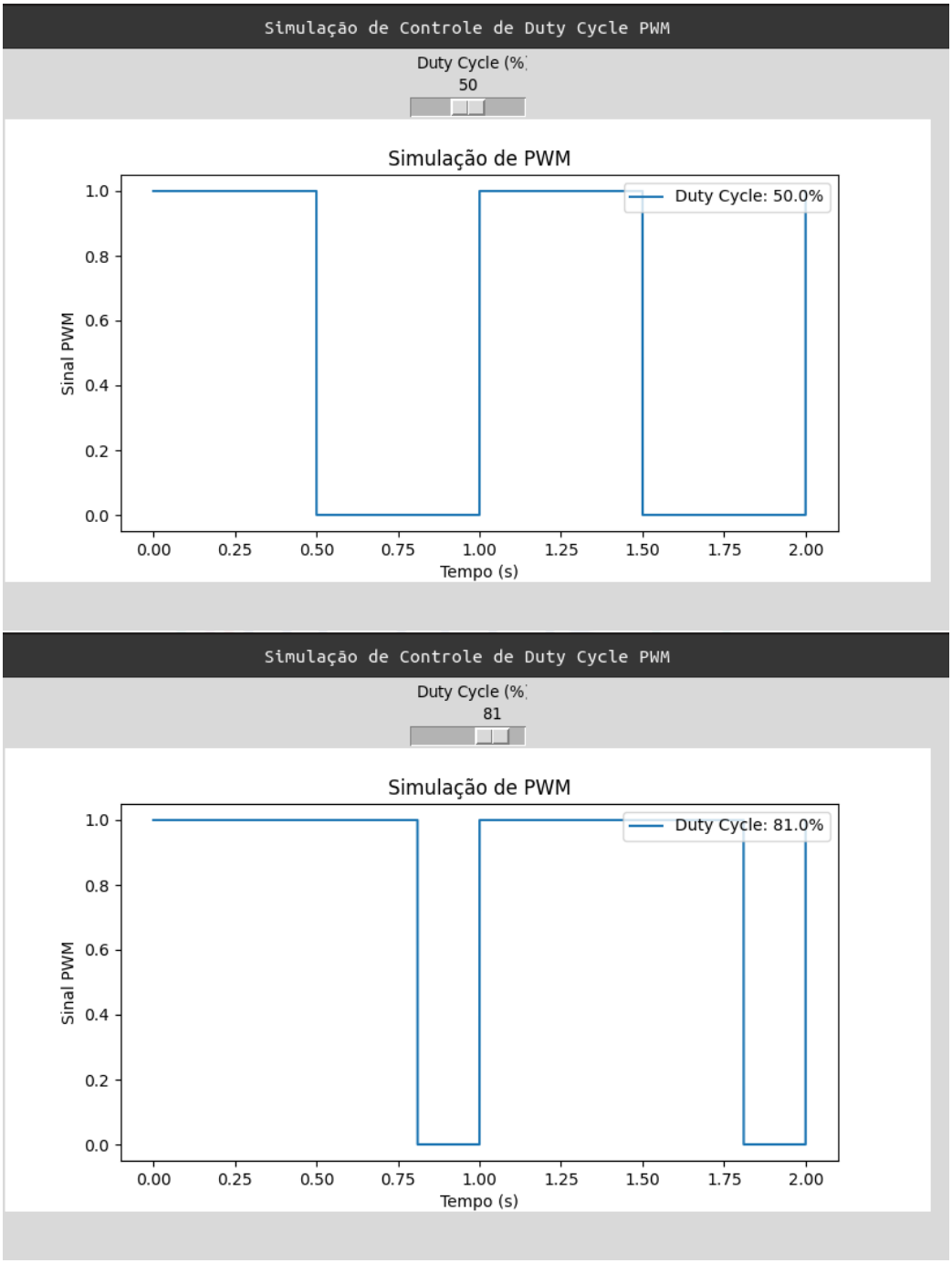


Figura 1: Representação do pulso PWM com duty cycles de 50% e 81%.

Observe que, conforme a equação 3, o tempo em que o sinal permanece desligado (inativo) é de 50% e 19%, respectivamente.

2.1.5 Duty Cycle no STM32

Após a compreensão dos conceitos básicos de PWM, o próximo passo é analisar como o microcontrolador STM32 interpreta esses ciclos. Para isso, é necessário entender o funcionamento do sistema de contagem do STM32, onde o valor do contador é comparado com o valor de comparação para determinar a porcentagem do duty cycle. Por exemplo, suponha que o contador tenha um valor máximo de 1000. Nesse caso, um duty cycle de 50% corresponderia a um valor de 500 no comparador. De forma análoga, um valor de 750 no comparador resultaria em um duty cycle de 75%. Esse valor de comparação é definido no timer, e enquanto o contador do timer não atingir esse valor, o pulso permanece ativo. Quando o contador atinge o valor de comparação, o pulso é desligado (inativo) até o reinício da contagem.

A seguir, é apresentada uma imagem que ilustra esse conceito.

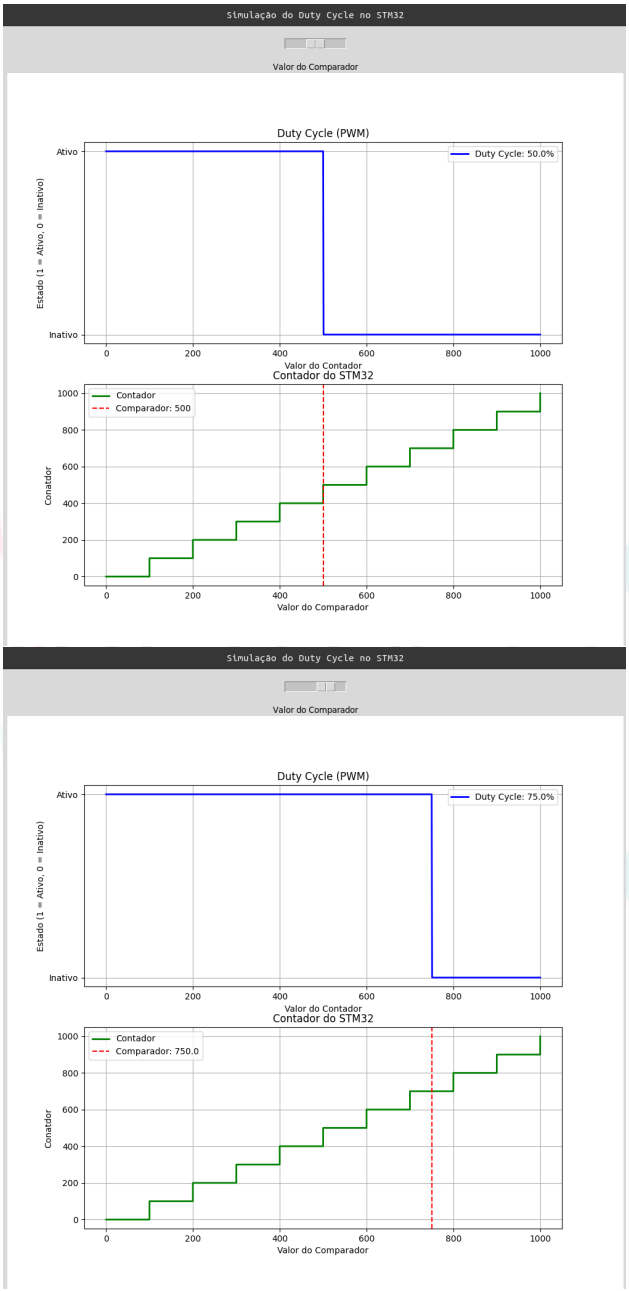


Figura 2: Ilustração do funcionamento do PWM no microcontrolador STM32, mostrando a relação entre o contador e o valor de comparação para determinar o duty cycle.

Esse conceito será de suma importância para o controle do braço robótico, uma vez que o *duty cycle* no controle do braço robótico define a quantidade de tempo que o sinal de controle permanece ativo em um ciclo de operação. O *duty cycle*, que é expresso como uma porcentagem, é diretamente responsável pela determinação da posição dos servomotores que controlam o movimento das articulações do braço. Quanto maior o *duty cycle*, maior será o deslocamento do servo, permitindo um controle preciso do movimento do braço.

A variação do *duty cycle* em cada servo motor resulta em diferentes posições, possibilitando a realização de tarefas complexas e a execução de movimentos específicos. Isso é crucial para a operação eficiente do braço robótico, garantindo que ele execute movimentos precisos e coordenados para realizar funções como pegar, mover ou posicionar objetos. O controle adequado do *duty cycle* permite uma resposta rápida e eficiente aos comandos, assegurando a eficácia do sistema robótico no cumprimento de suas tarefas.

2.1.6 Interrupção

interrupção em um microcontrolador é um mecanismo que permite pausar temporariamente a execução do programa principal para atender eventos de maior prioridade. Esses eventos podem ser originados por fontes externas, como sinais provenientes de sensores ou botões, ou internas, como timers, periféricos ou condições específicas detectadas pelo microcontrolador.

Quando uma interrupção ocorre, o microcontrolador registra sua origem, salva o estado atual do programa (incluindo o contador de programa e os registradores) e desvia para uma rotina específica chamada de ISR (Interrupt Service Routine). Essa rotina contém o código necessário para lidar com o evento que gerou a interrupção. Após concluir a execução da ISR, o microcontrolador restaura o estado anterior e retoma a execução do programa principal exatamente de onde foi interrompido.

As interrupções podem ser configuradas para disparar em condições específicas, como bordas de subida ou descida de um sinal, ou enquanto o evento que as gerou permanecer ativo. Além disso, há interrupções que podem ser habilitadas ou desabilitadas pelo software (máscaráveis) e outras que possuem alta prioridade e não podem ser desabilitadas (não-máscaráveis).

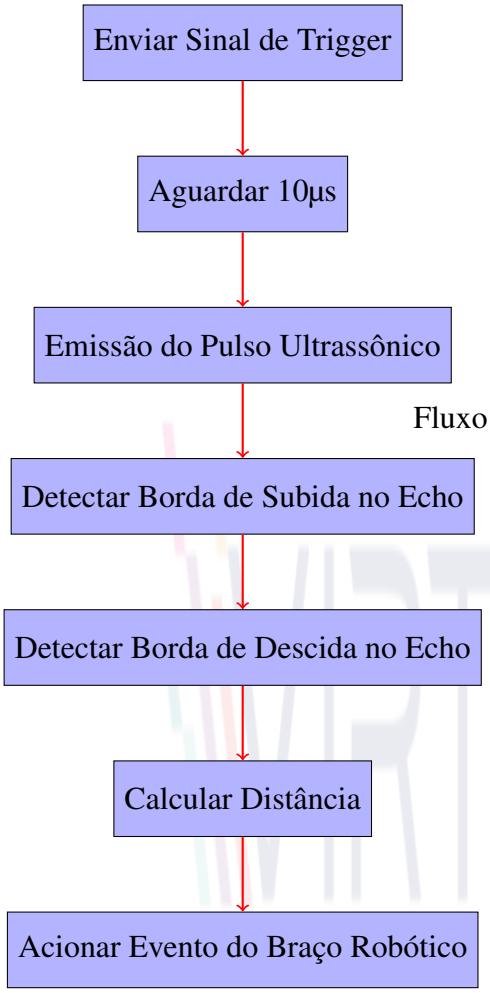
O microcontrolador utiliza esquemas de prioridade para determinar qual interrupção atender primeiro caso várias ocorram simultaneamente. Cada interrupção está associada a um vetor que aponta para o endereço da ISR correspondente.

Neste Projeto foi utilizado para junto com o sensor ultrasônico,conseguir detectar a distância do produto para acionar o braço robotico.

Na documentação, é mencionado que é necessário enviar um sinal de *trigger* e aguardar 10 microssegundos até o próximo pulso alto. Após esse tempo, ocorre uma borda de subida no canal do *echo*. Nesse caso, ativamos uma interrupção para detectar esse sinal de entrada e, em seguida, aguardamos a próxima mudança, que ocorrerá quando o sensor detectar um objeto e o pino *echo* voltar ao nível baixo.

Se detectarmos essa borda de descida, teremos o tempo de duração do pulso, que é proporcional à largura do sinal. Considerando um movimento linear e a velocidade do som no ar de 340 m/s, o sinal viaja até o objeto e retorna ao sensor, o que implica em um tempo total de ida e volta. Portanto, é necessário dividir o tempo por dois e multiplicá-lo pela velocidade do som para calcular a distância aproximada.

Esse processo nos fornece uma estimativa da distância entre o sensor e o objeto detectado, que pode ser utilizada, por exemplo, no cálculo da posição de um braço robótico, como no caso do acionamento de um evento relacionado a um movimento do braço.



Fluxo de Medição de Distância Ultrassônica

Figura 3: Diagrama de Blocos do Processo de Medição de Distância com HC-SR04

2.1.7 Comunicação I2C

O protocolo I2C (Inter-Integrated Circuit) é um padrão de comunicação serial síncrona desenvolvido para interconectar múltiplos dispositivos eletrônicos utilizando apenas duas linhas: uma linha de dados (SDA) e uma linha de clock (SCL). Essa arquitetura simplificada possibilita a comunicação eficiente entre um dispositivo mestre e um ou mais dispositivos escravos em um barramento compartilhado.

No barramento I2C, as linhas SDA e SCL são bidirecionais, controladas principalmente pelo dispositivo mestre. Todos os dispositivos conectados ao barramento compartilham essas duas linhas em um funcionamento half-duplex, ou seja, a comunicação ocorre em ambas as direções, mas não simultaneamente. O dispositivo mestre é responsável por iniciar, controlar e encerrar a comunicação, além de regular a linha de clock (SCL). Já os dispositivos escravos aguardam comandos do mestre e respondem de acordo com as instruções recebidas.

Cada dispositivo escravo no barramento possui um endereço único, utilizado para identificá-lo durante a comunicação. O processo de transmissão de dados começa quando o mestre gera uma condição de start, que ocorre ao puxar a linha SDA para o nível baixo enquanto a linha SCL permanece em nível alto. Em seguida, o mestre envia um byte de endereço do escravo, acompanhado por um bit de controle que indica se a operação será de leitura (R) ou escrita (W). Os dispositivos escravos monitoram o barramento e, ao reconhecerem seu endereço, respondem com um sinal de confirmação (ACK), indicando que estão prontos para a comunicação.

Após a transmissão de cada byte de dados, o receptor — seja o mestre ou o escravo — envia um sinal de reconhecimento (ACK), confirmando que o dado foi recebido corretamente. Caso o endereço ou a operação não sejam reconhecidos, um sinal de não reconhecimento (NACK) é enviado. A comunicação segue com a troca de bytes de dados, sendo o ACK emitido ao final de cada byte para confirmar o sucesso da transmissão.

Para encerrar a comunicação, o mestre envia uma condição de stop, que ocorre quando a linha SDA é liberada (vai para o nível alto) enquanto a linha SCL também permanece em nível alto. Esse sinal indica o término da transmissão e libera o barramento para futuras comunicações.Neste projeto, foi utilizada a comunicação com um dispositivo LCD para documentar os estados do braço robótico.

3 Projeto Final

3.1 Introdução ao Projeto

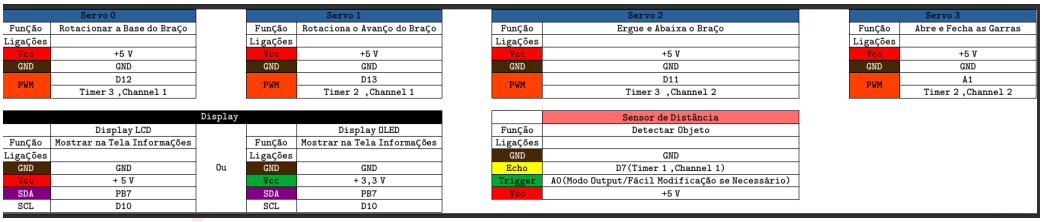
O projeto da garra robótica tem como objetivo automatizar tarefas repetitivas no ambiente industrial, com foco na otimização da distribuição logística de uma empresa. Essa automação visa aumentar a eficiência e reduzir erros humanos, promovendo um fluxo de trabalho mais rápido e preciso. Além disso, o projeto contempla o desenvolvimento futuro de sistemas de sensores avançados para a classificação e catalogação de produtos, garantindo uma gestão mais organizada e inteligente do inventário.

Os displays do sistema desempenharão um papel crucial ao exibir informações em tempo real sobre os objetos a serem transportados, como peso, dimensões, e status do processo de manipulação. Essas informações ajudarão os operadores a monitorar e controlar o trabalho da garra robótica com maior precisão. Além disso, será possível visualizar a situação atual da garra, como sua posição, velocidade e a tarefa em execução, o que facilita a tomada de decisões e a supervisão contínua.

Esses recursos visam não apenas otimizar o desempenho logístico, mas também aumentar a segurança no ambiente de trabalho. Com as informações do sistema acessíveis de maneira clara e intuitiva, será possível evitar possíveis acidentes, oferecendo maior controle e monitoramento das atividades. A implementação desses sistemas contribuirá para a criação de um ambiente de trabalho mais seguro, eficiente e tecnologicamente avançado, onde a interação entre homem e máquina se torna cada vez mais harmoniosa e eficaz.

3.2 Montagem e Integração

Após o desenvolvimento do algoritmo e a definição do escopo do projeto, é necessário utilizar a modularização detalhada nos algoritmos para sua implementação e integração, a fim de montar o projeto final. Abaixo, seguem os diagramas de montagem relacionados às pinagens.



Servo 0	
Função	Rotacionar a Base do Braço
Ligações	
Vcc	+5 V
GND	GND
PWM	D12
	Timer 3 ,Channel 1

Servo 1	
Função	Rotaciona o Avanço do Braço
Ligações	
Vcc	+5 V
GND	GND
PWM	D13
	Timer 2 ,Channel 1

Servo 2	
Função	Ergue e Abaixa o Braço
Ligações	
Vcc	+5 V
GND	GND
PWM	D11
	Timer 3 ,Channel 2

Servo 3	
Função	Abre e Fecha as Garras
Ligações	
Vcc	+5 V
GND	GND
PWM	A1
	Timer 2 ,Channel 2

Display			
	Display LCD		Display OLED
Função	Mostrar na Tela Informações		Função	Mostrar na Tela Informações
Ligações			Ligações	
GND	GND	Ou	GND	GND
Vcc	+ 5 V		Vcc	+ 3,3 V
SDA	PB7		SDA	PB7
SCL	D10		SCL	D10

	Sensor de Distância
Função	Detectar Objeto
Ligações	
GND	GND
Echo	D7(Timer 1 ,Channel 1)
Trigger	A0(Modo Output/Fácil Modificação se Necessário)
Vcc	+5 V

Figura 4: Esquematico das Ligações dos Pinos

Após a implementação e superação da etapa de codificação, foi necessário realizar ajustes finos para a calibração do sensor de distância, assim como para o braço robótico, a fim de garantir a execução completa dos movimentos. Como mencionado anteriormente nos capítulos, a tensão de 5V não será ideal, o que resulta no fato de que o servo motor não conseguirá completar o ângulo exato de 180 graus. O ângulo calculado foi de aproximadamente 120°.

Diante disso foi possível obter a seguinte figura

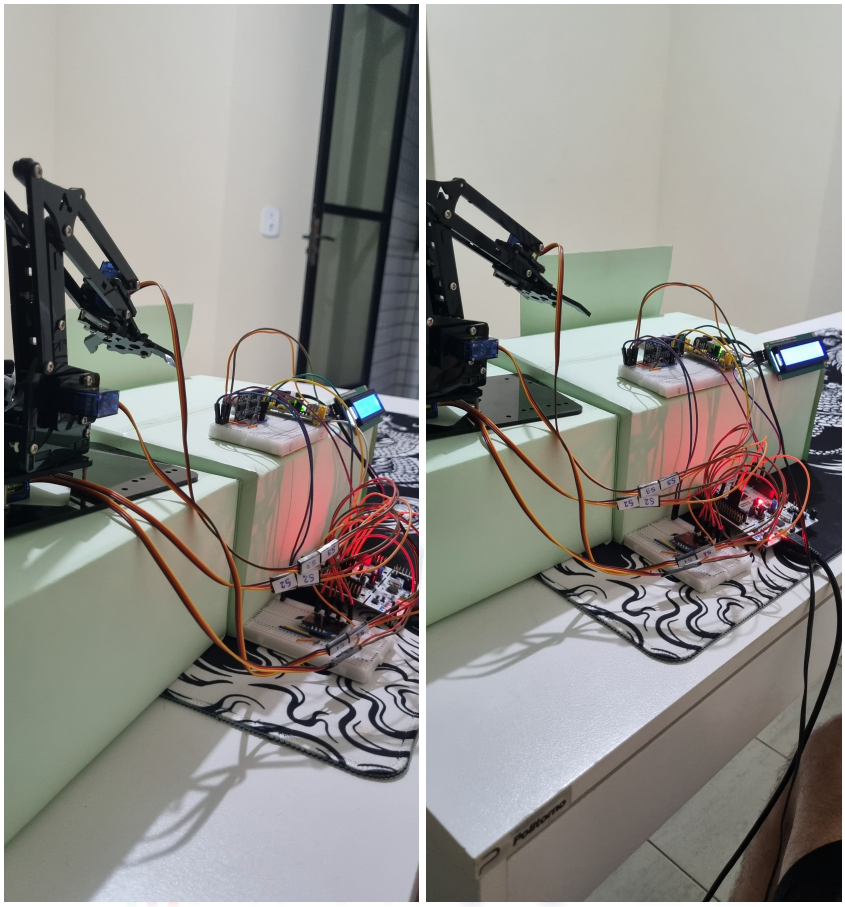


Figura 5: Montagem do Projeto

Após a definição de cada parte do código, integramos os módulos para que executassem as funções para as quais foram destinados. Com isso, obtivemos êxito, e os resultados podem ser visualizados abaixo.

Vale ressaltar que, para simular as informações dos produtos a serem transportados, utilizamos um vetor de strings contendo essas informações, que foram obtidas a partir de um banco de dados. No entanto, essa implementação poderia ter sido realizada utilizando um cartão microSD.

Quanto à classificação e separação dos produtos, uma proposta futura seria a implementação de cartões RFID, sensores de loteamento, como códigos de barras, ou até mesmo sensores de cor. O intuito deste projeto foi automatizar algumas atividades repetitivas, e sua escalabilidade poderia ser ampliada para o transporte de objetos maiores ou mais pesados em cenários reais.

Referências

[1] STMicroelectronics. *STM32F446xx Advanced ARM-Based 32-Bit MCUs*, n.d. Datasheet DM00135183.

[2] STMicroelectronics. Stm32f4 hal - hardware abstraction layer, 2023. Acesso em: 18 Dez. 2024.