Robô Explorador

SISTEMAS OPERACIONAIS EMBARCADOS

Filipe Alves de Sousa (15/0125429), Thompson Moutinho do Amaral Silva (12/0023245)
Engenharia Eletrônica
Faculdade Gama - Universidade de Brasília
Gama, DF
E-mail: fylypew@gmail.com, thompson.masilva@gmail.com

RESUMO

Este projeto tem como propósito a implementação de um dispositivo capaz de auxiliar no monitoramento de lugares de difícil acesso utilizando-se uma placa de desenvolvimento RaspBerry Pi, um dispositivo que proporciona facilidade, versatilidade e liberdade de experimentação.

I. INTRODUÇÃO

A robótica pode ser definida como a arte de projetar e aplicar robôs ou sistemas robóticos em empreendimentos humanos. [3]

A fim de proporcionar maior comodidade e segurança para as pessoas, muitas projetos voltados à robótica e automação vem sendo estudados e aplicados de forma a promover um acentuado e notório desenvolvimento tecnológico.

Isso pode ser verificado no uso de equipamentos controlados à distância, de forma precisa, possibilitando operações sofisticadas nas mais diversas tarefas.

Muitos dispositivos auxiliam na contínua análise e exploração de locais que carecem de inspeções e de reparo, tais como tubulações subterrâneas, lajes, dutos, entre outros. [1]

Portanto, devido à grande demanda por examinar ambientes de difícil acesso, foi proposta a implementação de um robô móvel capaz de transmitir imagens em tempo real, medir a temperatura e de ser controlado remotamente, através de uma comunicação sem fio, fazendo-se uso de uma plataforma de desenvolvimento Raspberry Pi e alguns módulos adicionais.

II. OBJETIVOS

Utilizando uma RaspBerry Pi, o objetivo desse projeto é desenvolver um dispositivo de fácil manuseio e baixo custo, cuja finalidade é empregá-lo em inspeções de lugares de difícil acesso, respeitando-se os limites tolerados pelos seus componentes em cada ambiente em questão.

III. REQUISITOS

O dispositivo deverá, de forma satisfatória, ser controlado à distância, via comunicação wifi e transmitir as imagens gravadas em tempo real:

Controle: Frente, trás, esquerda, direita e parado, através de um notebook ou um smartphone.

Transmissão de imagens: Filmar e transmitir, em tempo real, o ambiente inspecionado, com um delay máximo de 5s.

Medir a temperatura: Transmitir, em tempo real, a temperatura aferida no ambiente inspecionado, com um delay máximo de 5s e tolerância de ±0.5°C.

IV. JUSTIFICATIVA

Quando se trata de obtenção de informações em tempo real, em ambientes de difícil acesso, os robôs exploradores mostram-se muito funcionais. Visto que eles permitem analisar diversos locais, sendo controlados à distância.

Existem muitos robôs que podem ser utilizados para inspecionar, porém, geralmente são equipamentos com maior complexidade e alto custo.

A motivação deste projeto consiste em complementar os dispositivos básicos de inspeção, já existentes, para que, havendo necessidade, possa-se comprar ou construir o seu robô explorador, empregando-se a plataforma Raspberry Pi.

V. BENEFíCIOS

O dispositivo a ser projetado será de fácil manuseio. O operador poderá controlar o dispositivo através do seu notebook ou smartphone e registrar as imagens do lugar inspecionado em tempo real, além de verificar a hora e a temperatura medida. Todo o processamento será realizado pela Raspberry Pi.

Desta forma, será viabilizada uma maior comodidade e praticidade ao operador, de forma a mantê-lo isento da exposição física aos efeitos do ambiente de difícil acesso em questão.

VI. REVISÃO BIBLIOGRÁFICA

Existem alguns projetos semelhantes ao proposto aqui. Um dos exemplos mais conhecidos é o Robô Seguidor de Linha. - É difícil não lembrar do frenético robô faxineiro do filme WALL-E, uma co-produção dos estúdios Walt Disney Pictures e Pixxar Animations Studios. [6] Nesse filme, o robozinho faxineiro percorre uma faixa - guia - através do qual deve limpar eventuais sujeiras que apareçam. Esta ideia não é muito distante do que já é possível fazer hoje em dia.

No projeto “Robô seguidor de linha com Raspberry Pi Zero W e OpenCV” [7] uma webcam é utilizada para se obter imagens de uma guia desenhada no chão. Há um processamento de imagens desta guia que controla o movimento do Robô utilizando uma linha central como referência, quando a linha está para a esquerda da guia o robô se movimenta para esquerda e quando a linha está para a direita o robô se movimenta para a direita. Feito isto, o robô controla seu percurso seguindo a linha guia.

Um outro projeto, utiliza miniaturas de carros guiados pela RaspBerry usando sensores ultra sônicos. Quando o carro chega perto de um obstáculo o sensor envia um sinal para a Rasp, esta por sua vez processa a informação e envia sinais para os servo-motores, automaticamente, permitindo a rotação das rodinhas, assim, o carro desvia dos obstáculos de maneira autônoma. [8]

VII. DESCRIÇÃO DE HARDWARE

Neste projeto, serão utilizados os seguintes componentes:

Tabela 1. Lista de materiais

Componente	Tipo / especificação	Quanti.
Raspberry Pi	Zero Wh	1
Cartão micro SD	16GB - Classe 10	1
Dissipador p/ RB	Alumínio	1
Fonte p/ RB	5V e 2A - DC	1
Ponte H	L298N	1
Motor DC com caixa de redução	3 - 12 V	2
Bateria p/ motores	9 V - DC	1
Câmera p/ RB	5MP - Ov5647 - com visão noturna	1
Cabo flat	p/ camera, RB pi zero	1
Fios	Jumpers	20
Termopar	Texas Instruments - ADS1118	1
Display LCD	NHD - C0216CZ-FSW - SPI	1
Rede em comum	Wifi - Wireless	1
Computador	Pessoal	1
Rodas	-	3
Estrutura	Chassi de acrílico	1

- **A Raspberry Pi Zero Wh:**

A Raspberry Pi Zero Wh é uma placa de baixo custo com um tamanho reduzido (65 x 30 x 5mm). Ela conta com WiFi e Bluetooth integrados, o seu processador é o Broadcom BCM2835 single-core de 1GHz que, junto à memória de 512MB, permite que aplicações usando 40 pinos de GPIO. Além disso, ela tem slot para cartão micro SD, conector CSI para câmera, conector de vídeo mini HDMI, 2 portas USB (1 para dados e outra para alimentação 5V) e roda diversas distribuições Linux, como o Raspbian e Ubuntu. A placa vem com a barra de pinos soldada e sua fonte de alimentação é de 5V/2A.

Figura 1. Raspberry Pi Zero Wh.

Com o intuito de cumprir os requisitos, o projeto foi descrito conforme o diagrama simplificado a seguir:

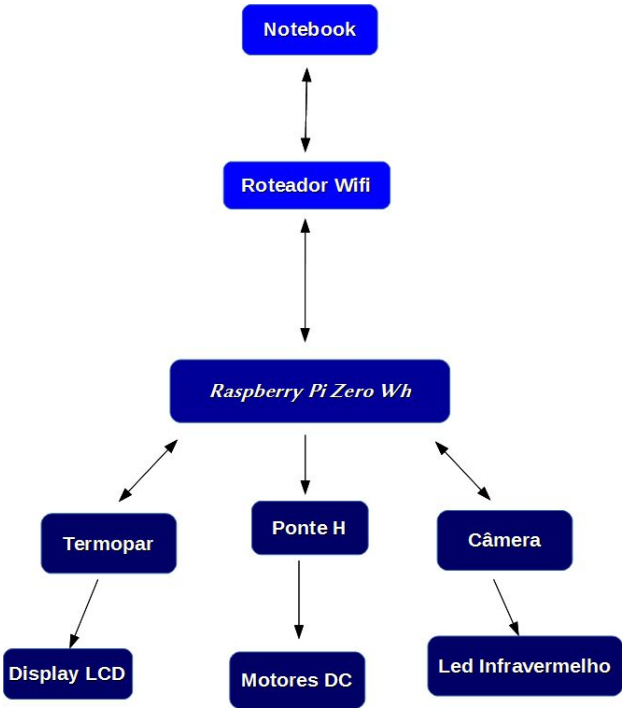


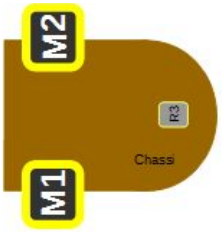
Figura 2. Diagrama de blocos funcional do projeto - fluxo de dados.

Neste projeto, através de um servidor local, a Raspberry proporciona, ao usuário, uma comunicação com três módulos principais, sendo eles: Controle dos motores, Medição de temperatura e Transmissão de imagens.

- **Controle dos motores DC:**

Visto que os motores DC têm como finalidade permitir o deslocamento do robô, para controlá-los, foi necessário estudar o seu funcionamento.

Durante o acionamento, os motores trabalham de forma coordenada. Assim, para realizar o deslocamento para frente, os dois motores rotacionam, simultaneamente, no sentido horário, para deslocar-se para trás, os motores giram no sentido anti-horário e, para ir para direita ou esquerda, um rotaciona em um sentido enquanto o outro rotaciona no sentido inverso. Conforme o esquema ilustrado abaixo:



	M1	M2
Frente	Horário	Horário
Trás	Anti-horário	Anti-horário
Direita	Horário	Anti-horário
Esquerda	Anti-horário	Horário

Figura 3. Controle de movimento do robô, de acordo com os sentidos de rotação, onde M1 e M2 são os motores controlados e R3 é a terceira roda.

A comutação do sentido de rotação do motor, foi realizada através de uma ponte H.

O driver empregado foi o L298N, um módulo composto por transistores e diodos de proteção, que possibilitam o chaveamento e a passagem de corrente elétrica na direção correta para cada caso.

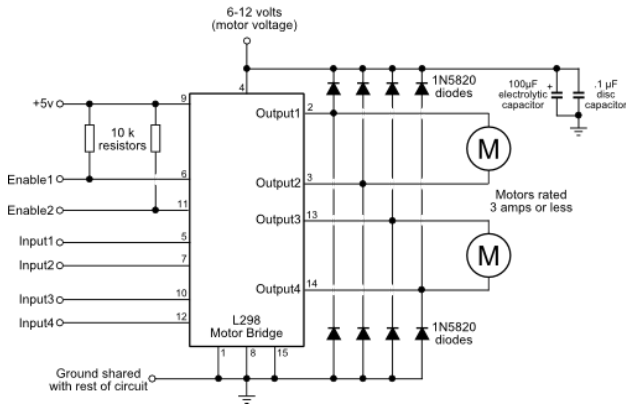


Figura 4. Esquemático de conexão dos componentes do módulo e motores ao CI L2898N.

Este módulo permite que as polaridades da alimentação fornecida aos motores sejam invertidas, fazendo com que eles girem no sentido horário e anti-horário.

Pode-se controlar dois motores DC, onde os pinos 5, 7, 10 e 12 são conectados aos - GPIOs - pinos de saída da Raspberry Pi, para a comutação dos motores.

O pino 9 é responsável pela alimentação do circuito, ligado em 5V. Enquanto os pinos 2 (motor 1) e 13 (motor 2) recebem uma entrada analógica de 0V a 5V para o controle de velocidade. Ambos foram conectados em +5V.

O pino 4 é responsável pela alimentação dos motores, podendo variar de 6V a 12V. Sendo ligado em 9V, com uma bateria.

Tabela 2. Pinagem de conexão - controle dos motores.

Ponte H - L298N		**RB Pi Zero Wh**	
4	VCC	-	-
5	INPUT1	7	GPIO PWM
7	INPUT2	11	GPIO PWM
10	INPUT3	13	GPIO PWM
12	INPUT4	15	GPIO PWM
-	GND	6	GND

- **Medição de temperatura:**

O SoC (System on a Chip), processador Broadcom BCM2835 ARM11, usado no Raspberry Pi Zero Wh, é equivalente à alguns chips usados em smartphones antigos. Ele opera na faixa dos 700 MHz e oferece um desempenho próximo dos 0.041 GFLOPS. Apesar de ter recursos gráficos que possibilitam um bom desempenho, em determinadas aplicações, ele não aquece muito. No entanto, pode aquecer bastante quando o uso de CPU chega próximo dos 100%.

Para melhorar o desempenho do SoC, é possível fazer um overclock, aumentando-se a frequência de operação. Isso provoca um aumento de temperatura do chip, que pode chegar a 85 °C. Nesse caso, pode ser necessário colocar um dissipador de calor de tamanho apropriado.

Visto que a temperatura do ambiente interfere de forma significativa no funcionamento e desempenho de placas de circuitos eletrônicos. Em um sistema onde a Raspberry Pi está sendo empregada, a temperatura central do chip, a carga da CPU, o ajuste dinâmico da velocidade de clock e a tensão do núcleo, são fatores que podem ser afetados pela temperatura do ambiente a qual tal sistema está inserido.

O desempenho do SoC é reduzido quando a demanda na CPU é baixa, ou quando está muito quente. E quando a CPU tem alta demanda e a temperatura do chip é aceitável, o desempenho é temporariamente aumentado. Quando o SoC detecta ocorrência de superaquecimento interno, ele diminui a sua taxa de clock, a fim de reduzir o aquecimento. A faixa de operação de temperatura do chip em questão é de -40 a 85 °C, entretanto, no projeto, deve-se levar em consideração os limites de temperatura do restante das placas, periféricos e da estrutura do robô.

Como o excesso de temperatura pode causar falhas e/ou danificar o hardware de forma irreversível, para garantir que o robô esteja operando em condições de temperatura suportada, um sensor de temperatura foi adicionado ao dispositivo, de forma que ele afere a temperatura do ambiente, enviando os dados obtidos ao usuário e permitindo o monitoramento em tempo real.

Optou-se por utilizar a placa de desenvolvimento 430BOOST-ADS1118 da Texas Instruments.

Essa placa possui um termopar com conversor ADS e um display 16x2. Além disso, ela é caracterizada

por operar na faixa de -40 a 125 °C, dispor de 16 bits de resolução, oferecer flexibilidade de modelagem em experimentos e apresentar uma resposta rápida e boa taxa de atualização.

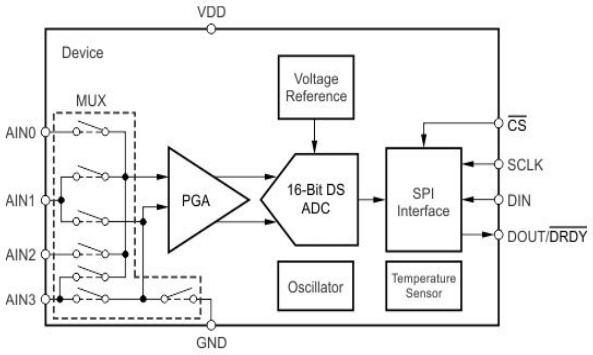


Figura 5. Diagrama funcional do ADS 1118.

A placa foi conectada ao Raspberry Pi, utilizando-se comunicação Serial Peripheral Interface (SPI). Foram necessários oito fios (jumper macho-fêmea).

O LCD tem duas linhas de 16 caracteres, onde decidiu-se usar uma das linhas para exibir a hora e a temperatura atual, medida pelo termopar. Além disso, o LCD permite ao usuário enviar mensagem para que pessoas próximas ao robô leiam de imediato. Por exemplo, pode-se dizer: "Não toque" ou "Teste nº 1".

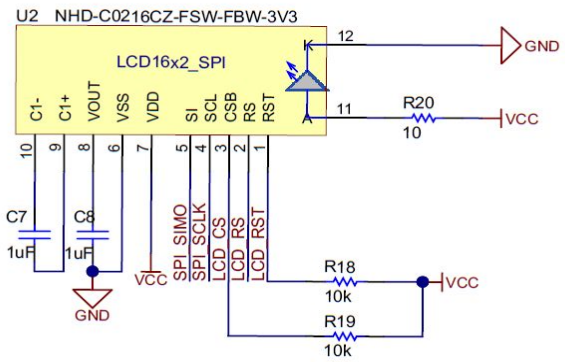


Figura 6. Esquemático do display 16x2.

Tabela 3. Pinagem de conexão - medição de temperatura.

Placa ADS1118		**RB Pi Zero Wh**	
P1.1	VCC	1	3.3 V
P1.7	CLK	23	CLK
P1.8	ADS_CS	26	SPI
P2.8	LCD_CS	24	SPI
P2.9	LCD_RS	11	GPIO17
P2.6	GND	9	GND
P2.7	SIMO	19	MOSI
P2.1	SOMI	21	MISO

As conexões descritas podem ser verificadas na imagem abaixo:

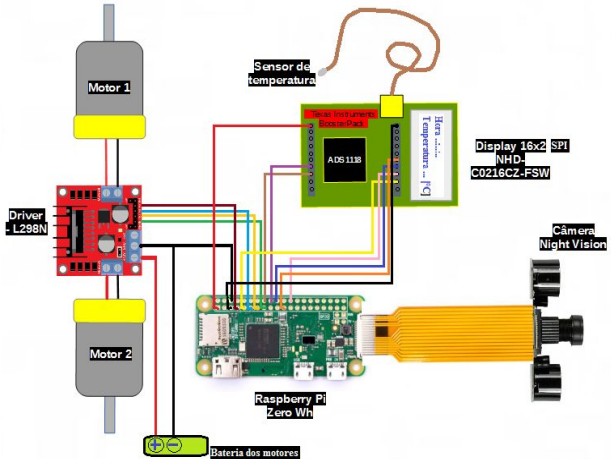


Figura 7. Esquemático de conexão unindo todos os módulos à Raspberry.

VIII. DESCRIÃO DE SOFTWARE

- **Controle dos motores DC:**

O código apresentado nos apêndices deste relatório é a base de funcionamento e controle dos motores que serão acoplados às rodas. Através dele que a movimentação do robô será implementada na RaspBerry e controlada pelo teclado do notebook ou smartphone. O código implementado em Python pode ser dividido, basicamente, em três partes: bibliotecas, controle e finalização. Conforme o diagrama abaixo:

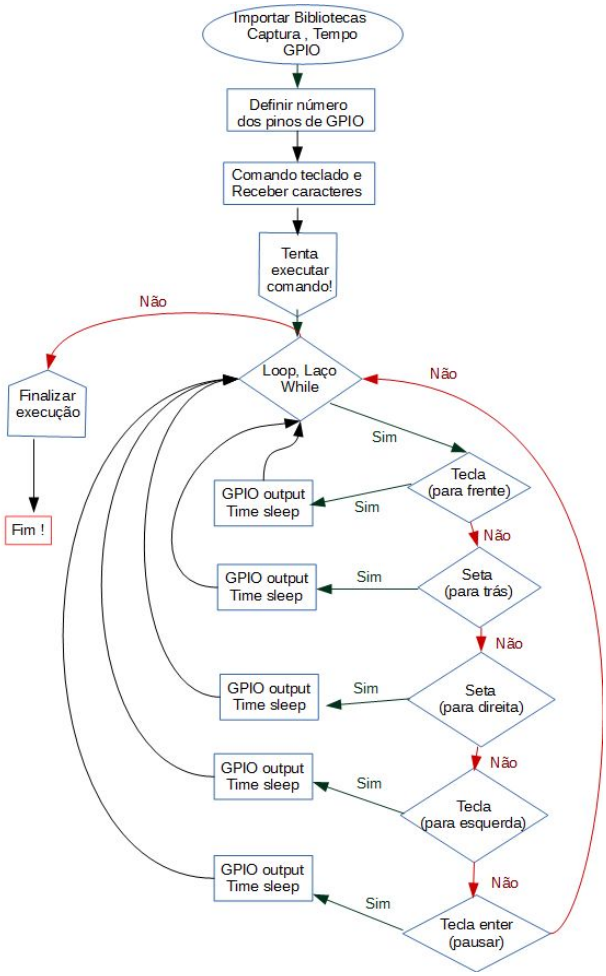


Figura 8. Fluxograma utilizado no desenvolvimento do script de controle dos motores.

A seguir comenta-se sobre cada uma dessas partes individualmente.

1) Bibliotecas e pinagem entrada e saída

```
# importando bibliotecas de captura - tempo e GPIO
import curses
import RPi.GPIO as GPIO
import time

# definindo a pinagem - GPIO - pinos de saída
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7,GPIO.OUT)
GPIO.setup(11,GPIO.OUT)
GPIO.setup(13,GPIO.OUT)
GPIO.setup(15,GPIO.OUT)
```

Figura 9. Primeira parte, bibliotecas e pinos.

Foram usadas três bibliotecas, Figura 6:

➔ Curses: fornece funcionalidades básicas como preparar o console, criar "janelas", escrever nessas janelas, receber caracteres.

➔ RPi.GPIO as GPIO: **G**eneral **P**urpose **I**nput **O**utput, são os pinos programáveis de entrada e saída da placa.

➔ Time: implementa um contador de tempo.

Abaixo das bibliotecas, foram definidos os pinos usados para entrada e saída que irão orientar para que lado o motor vai girar.

```
screen = curses.initscr()
curses.noecho()
curses.cbreak()
screen.keypad(True)

try:
        while True:
            char = screen.getch()
            if char == ord('q'):
                break
            elif char == curses.KEY_UP:
                GPIO.output(7,False)
                GPIO.output(11,True)
                GPIO.output(13,False)
                GPIO.output(15,True)
            elif char == curses.KEY_DOWN:
                GPIO.output(7,True)
                GPIO.output(11,False)
                GPIO.output(13,True)
                GPIO.output(15,False)
            elif char == curses.KEY_RIGHT:
                GPIO.output(7,True)
                GPIO.output(11,False)
                GPIO.output(13,False)
                GPIO.output(15,True)
            elif char == curses.KEY_LEFT:
                GPIO.output(7,False)
                GPIO.output(11,True)
                GPIO.output(13,True)
                GPIO.output(15,False)
            elif char == ord('d'):
                GPIO.output(11,True)
                GPIO.output(15,True)
                time.sleep(.5)
                GPIO.output(7,True)
                GPIO.output(11,False)
                GPIO.output(13,True)
                GPIO.output(15,False)
                time.sleep(.5)
                GPIO.output(7,True)
                GPIO.output(11,False)
                GPIO.output(13,False)
                GPIO.output(15,True)
                time.sleep(.5)
                GPIO.output(7,False)
                GPIO.output(11,True)
                GPIO.output(13,True)
                GPIO.output(15,False)
                time.sleep(.5)
                GPIO.output(11,False)
                GPIO.output(13,False)
            elif char == 10:
                GPIO.output(7,False)
                GPIO.output(11,False)
                GPIO.output(13,False)
```

Figura 10. Segunda parte, controle direcional.

Na Figura 7. temos o código que controla os movimentos e direções que o motor realiza. O motor tem a opção de 4 movimentos, para cima, para baixo, para direita e para esquerda.

```
finally:
    # finalizar, fechar o cursor corretamente, inc,
    # ligue o eco novamente!
    curses.nocbreak(); screen.keypad(0); curses.echo()
    curses.endwin()
    GPIO.cleanup()
```

Figura 11. Terceira parte, finalização.

A terceira parte do código é a finalização e é apresentado na figura 11. O programa encerra o cursor e liga o eco se desejado.

- **Medição de temperatura:**

O código em C usado para aquisição de temperatura em tempo real através do termopar é extenso. Uma síntese de descrição do código em C é apresentada a seguir. O código é usado para recuperar as medições de temperatura realizadas pelo sensor e faz conexões com a RaspBerry.

O inicio do código aplica bibliotecas e define variáveis globais para toda a programação. Em seguida são definidas funções globais para serem utilizadas no código C principal. A seguir uma descrição resumida das principais funções do código C principal.

- *delay_ms* usada para pausar por um curto período de tempo;
- *setup_io* usada para alocar memória;
- *lcd_writecom*, *lcd_writedata*, *lcd_clear, cd_display_string, lcd_init* são comandos para funcionalidade do LCD como leitura e escrita, inicialização, etc;
- *therm_transact* envia quatro bytes, dois bytes de configuração enviados duas vezes e retorna dois bytes;
- *local_compensation* transforma o código de temperatura interna para um código de compensação de temperatura que é adicionado ao código do sensor termopar;
- *adc_code2temp* converte os resultados do conversor AD para temperatura em uma faixa de -270 à 500 °C;
- *ads_config* configura e inicia a conversão;
- *ads_read* ler o resultado do conversor AD e inicia uma nova conversão;
- *get_measurement* retorna a temperatura medida;
- *get_measurement_fast* ler a medida do sensor de temperatura externo e reinicia o sensor externo;

O código em C principal faz a comunicação entre os sensores de temperatura, a raspberry e o display de LCD iniciando-se com a função main(). As primeiras linha de código são definições de variáveis locais utilizadas no código C principal. Os botões de entrada e saída GPIO são os primeiros a serem chamados pelas funções *setup_io(), INP_GPIO() e OUT_GPIO()*. Em seguida a lógica para análise das entradas são realizadas. A função *if (argc>2)* imprime uma mensagem no display e uma mensagem de saída se nece. A função de lógica *if (argc>1)* inicializa o display de LCD e imprime o período decorrido entre cada medição de temperatura além de uma mensagem pré estabelecida pelo programador. Também possui lógica para encerrar o programa e imprimir uma mensagem de "despedida" caso necessário. O laço *if (arcg>3)* imprime no LCD as medidas de temperatura e uma mensagem de despedida.

A parte seguinte do código permite abrir SPI para o termopar, faz uma única medição de temperatura, imprime logo em seguida e retorna para o início do laço. Desta forma as temperaturas podem ser obtidas a cada 1 segundo de medição do termopar.

IX. RESULTADOS

A primeira parte foi implementada e alimentada adequadamente, conectando-se os respectivos pinos definidos no código à Raspberry, de modo que a ponte H receba os sinais de comutação para controlar os motores.

Com o Sistema Operacional Raspibian Jessie já instalado, através da rede de comunicação, entre o servidor e usuário, foi possível executar o script desenvolvido em Python, verificando-se o perfeito funcionamento dos comandos.

Utilizando um computador pessoal, o usuário, ao clicar na seta para cima, os motores rotacionam, simultaneamente, no sentido horário; Clicando para baixo, eles giram no sentido anti-horário; Para a esquerda ou direita, eles giram em sentidos contrários; E, teclando "Enter", os motores param.

Implementou-se também a obtenção da temperatura ambiente ao redor do robô através de um termopar acoplado à RaspBerry. Este sensor produz medidas de temperatura a cada segundo e apresenta os

dados no terminal de programação e também no display LCD acoplado ao robô. A figura 12 apresenta os resultados obtidos impressos no terminal de programação enquanto que a figura 13 apresenta o resultado exibido no display de LCD acoplado.

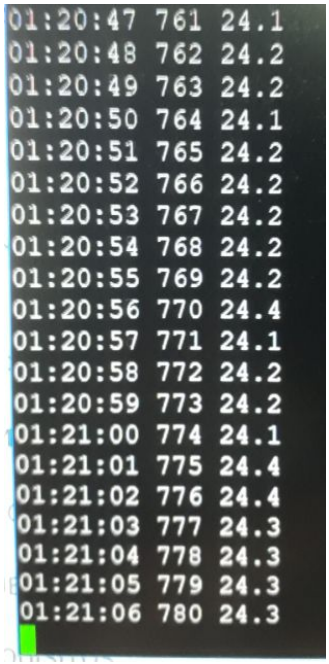


Figura 12. Resultados obtidos a cada segundo pelo sensor de temperatura.

O algoritmo implementado a cada segundo lê o sensor de temperatura interna uma vez e o termopar externo dez vezes em um curto tempo (algumas centenas de milissegundos no total) para que as medições possam ser calculadas em média e finalmente enviadas para uma resolução de 0,1 graus. O resultado final foi muito bom.

Figura 13. Resultado da medida de temperatura no horário exibido.

Observar que na figura 12 temos o horário de medição realizado, o número correspondente à medida e a temperatura em °C obtida. Uma comparação entre as figuras 12 e 13 observa-se que a medida número 768 realizada às 01:20:58 apresentou 24,2°C que é o mesmo resultado mostrado no display de LCD acoplado ao robô explorador.

Na figura 14 pode-se verificar o projeto implementado ainda de forma parcial.

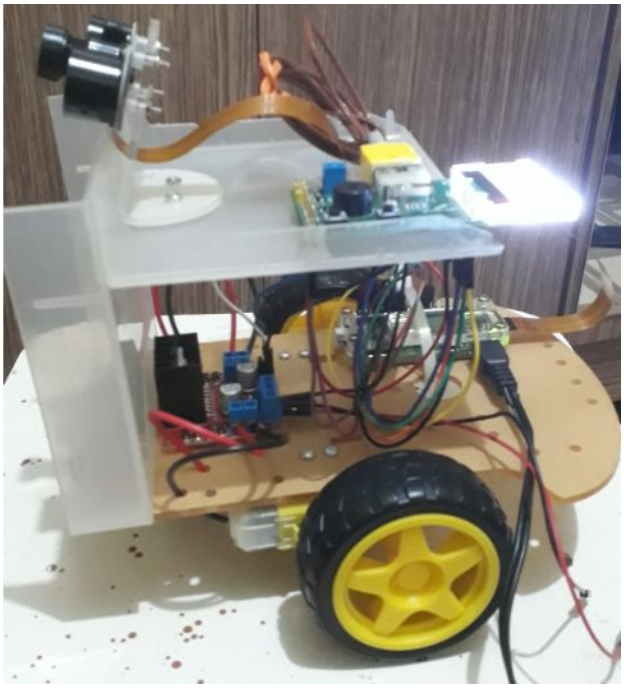


Figura 14. Robô explorador implementado até o momento.

X. CONCLUSÃO

Como o robô explorador está em fase de desenvolvimento, até o presente momento, apenas as partes do controle dos motores e de monitoramento de temperatura foram implementadas.

Verificou-se que a Raspberry Pi é uma plataforma de desenvolvimento satisfatória em projetos de controle e de monitoramento.

Pode-se concluir que todo componente tem um limite de temperatura no qual pode operar. Overclocks extremos e ou locais com altas temperaturas podem fazer com que a temperatura máxima de um ou mais componentes ultrapassem o limite e sofram danos irreversíveis.

O aquecimento das placas, especialmente da Raspberry Pi, pode ser um fator preocupante, entretanto, sabe-se que o firmware da Raspberry Pi está programado para começar a diminuir a frequência da CPU, quando chega-se à 80° graus, para consequentemente diminuir a temperatura. Isso acontece gradativamente, mas ao atingir uma temperatura maior ou igual a 85°C, o clock é reduzido, diminuindo-se o desempenho. Nesse sentido, levando-se em consideração as diferenças entre um chip e outro, a temperatura do ambiente, pode influenciar no comportamento do sistema.

Tendo em vista que em alguns experimentos de inspeção, pode ser necessário verificar a temperatura do ambiente à distância, com um sensor de temperatura conectado ao Raspberry Pi, foi possível implementar essa parte do dispositivo e obter a temperatura do local inspecionado, verificando-se os valores em tempo real. Isso permite o uso dessas informações para estudos diversos e para evitar eventuais danos ao sistema, causados pela temperatura do ambiente.

Foi complicado e trabalhoso fazer interface do sensor usado (um termopar ADS1118). Seu o amplificador é um tanto complexo, já que que é capaz de medir uma tensão extremamente pequena. Este booster veio com firmware, disponibilizado pela TI destinado ao dispositivo MSP430G2553. O software também fornece código-fonte comentado para os usuários começarem a desenvolver aplicativos. Assim, o código que a TI disponibiliza, foi alterado e adaptado para ser implementado na Raspberry pi Zero W, usando suas entradas / saídas (I / O).

Desta forma, após muitas tentativas e análise do comportamento e funcionamento dos componentes empregados no robô explorador, foi possível implementar conforme o esperado.

XI. REFERÊNCIAS

[1] PAZOS, F. Automação de Sistemas e Robótica. Rio de Janeiro: Axcel, 2000.

[2] Instructables, Alvaro.Palero. Robot Arduino Explorator “Nueve”. Disponível em:<https://www.instructables.com/id/Robot-Arduino-Explorador-Nueve/>Acessado em: 25 de março de 2019.

[3] Joildo Schuerof. DESENVOLVIMENTO DE UM SISTEMA DE VISÃO ARTIFICIAL PARA UM ROBO EXPLORADOR. Disponível em: <http://larm.ufsc.br/files/2013/04/TCC-Joildo.pdf> Acessado em: 26 de março de 2019.

[4] Matt Richardson Shawn Wallace. Primeiros Passos com o Raspberry Pi. Disponível em:<http://www.martinsfontespaulista.com.br/anexos/produtos/capitulos/704370.pdf>Acessado em: 27 de março de 2019.

[5] Raspberrypi, Raspbian Stretch with desktop and recommended software. Disponível em:<https://www.raspberrypi.org/downloads/raspbian/> Acessado em: 28 de março de 2019.

[6] *WALL·E,* Walt Disney Pictures e Pixxar Animations Studios. Disponível em: <https://pt.wikipedia.org/wiki/WALL%C2%B7E>Acessado em: 27 de março de 2019.

[7] "Robo seguidor de linha com Raspberry Pi Zero W e OpenCV". Disponível em: <https://www.filipeflop.com/blog/robo-seguidor-de-linha-pi-zero-w-opencv/> Acessado em: 27 de março de 2019.

[8] "Protótipo de carro desenvolvido em RaspBerry Pi", TCC Faculdade de Pindamonhangaba, Autores: Evandro Barbosa e Gabriel Santos. Disponível em: <http://www.bibliotecadigital.funvicpinda.org.br:8080/jspui/bitstream/123456789/628/1/BarbosaSantos.pdf> Acessado em: 28 de março de 2019.

[9] DUAL FULL - BRIDGE DRIVER. Datasheet. Disponível em: <https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf> Acessado em: 23 de abril de 2019.

[10] Using the L298 Motor Bridge IC. Disponível em : <http://www.robotoid.com/appnotes/circuits-l298-hbridge.html> Acessado em: 25 de abril de 2019.

[11] Texas Instruments. ADS1118. Disponível em : <http://www.ti.com/product/ADS1118> Acessado em: 10 de maio de 2019.

[12] RPi Low-level peripherals. Disponível em : <https://elinux.org/RPi_Low-level_peripherals#C_2> Acessado em: 15 de maio de 2019.

[13] Texas Instruments. ADS1118 BoosterPack. Disponível em : <http://www.ti.com/tool/430boost-ads1118>Acessado em: 18 de maio de 2019.

[14] Ever Pi. Tudo sobre Raspberry Pi. Disponível em : <http://blog.everpi.net/2017/04/raspberry-pi-zero-overlock-extremo-1600mhz.html>Acessado em: 24 de maio de 2019.

[15] Shabaz Yousaf, BBB - FPGA / CPLD Programmer for the BeagleBone Black, Element 14 Tutorials, October 11, 2013.

[16] Xillinx, Spartan-6 Libraries Guide for HDL. Disigns, Xilinx user guide, April 24, 2012.

APÊNDICE

Código para controle dos motores

```
# importando bibliotecas de captura - tempo e GPIO
import curses
import RPi.GPIO as GPIO
import time

# definindo a pinagem - GPIO - pinos de saída
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7,GPIO.OUT)
GPIO.setup(11,GPIO.OUT)
GPIO.setup(13,GPIO.OUT)
GPIO.setup(15,GPIO.OUT)

# Apresentando a janela de dados, desativar o retorno
# do teclado para ativar a tela
# Resposta teclada em tempo real (sem espera)
# usando valores especiais nas teclas do cursor
screen = curses.initscr()
curses.noecho()
curses.cbreak()
screen.keypad(True)

try:
    while True:
        char = screen.getch()
        if char == ord('q'):
            break
        elif char == curses.KEY_UP:
            GPIO.output(7,False)
            GPIO.output(11,True)
            GPIO.output(13,False)
            GPIO.output(15,True)
        elif char == curses.KEY_DOWN:
```

```
            GPIO.output(7,True)
            GPIO.output(11,False)
            GPIO.output(13,True)
            GPIO.output(15,False)
        elif char == curses.KEY_RIGHT:
            GPIO.output(7,True)
            GPIO.output(11,False)
            GPIO.output(13,False)
            GPIO.output(15,True)
        elif char == curses.KEY_LEFT:
            GPIO.output(7,False)
            GPIO.output(11,True)
            GPIO.output(13,True)
            GPIO.output(15,False)
        elif char == ord('d'):
            GPIO.output(11,True)
            GPIO.output(15,True)
            time.sleep(.5)
            GPIO.output(7,True)
            GPIO.output(11,False)
            GPIO.output(13,True)
            GPIO.output(15,False)
            time.sleep(.5)
            GPIO.output(7,True)
            GPIO.output(11,False)
            GPIO.output(13,False)
            GPIO.output(15,True)
            time.sleep(.5)
            GPIO.output(7,False)
            GPIO.output(11,True)
            GPIO.output(13,True)
            GPIO.output(15,False)
            time.sleep(.5)
            GPIO.output(11,False)
            GPIO.output(13,False)
        elif char == 10:
            GPIO.output(7,False)
            GPIO.output(11,False)
            GPIO.output(13,False)
            GPIO.output(15,False)

finally:
    # finalizar, fechar o cursor corretamente, inc, ligue o
eco novamente!
    curses.nocbreak(); screen.keypad(0); curses.echo()
    curses.endwin()
    GPIO.cleanup()
```