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:

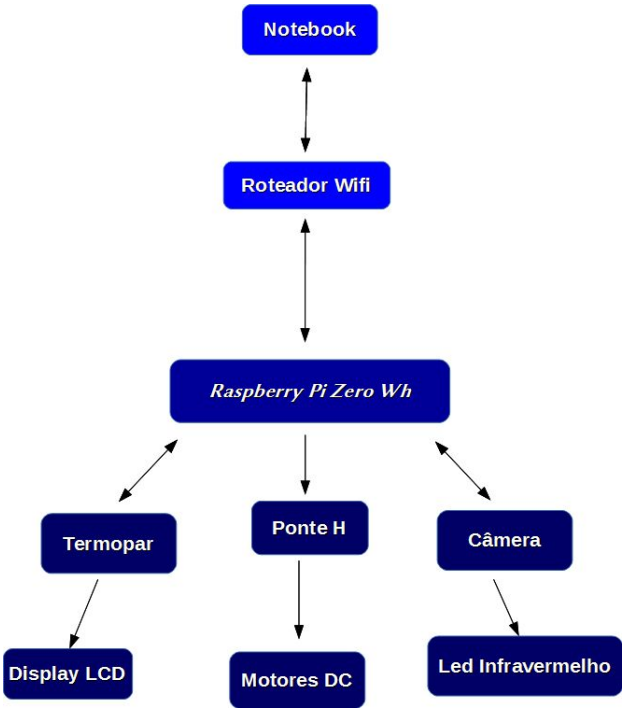


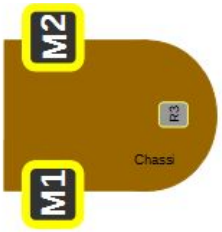
Figura 2. Diagrama de blocos funcional do projeto - fluxo de dados.

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.

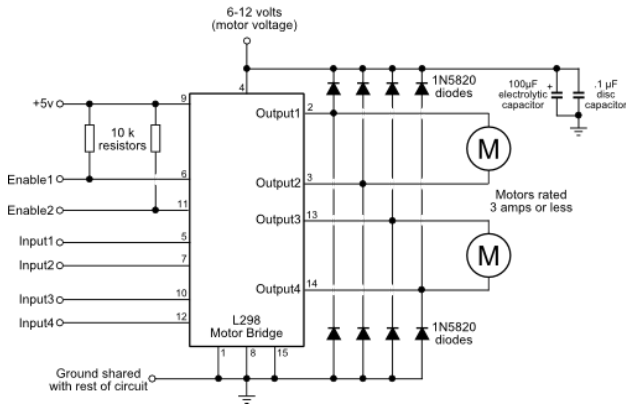


Figura 4. Esquemático de conexão dos componentes do módulo e motores ao CI L2898N.

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.

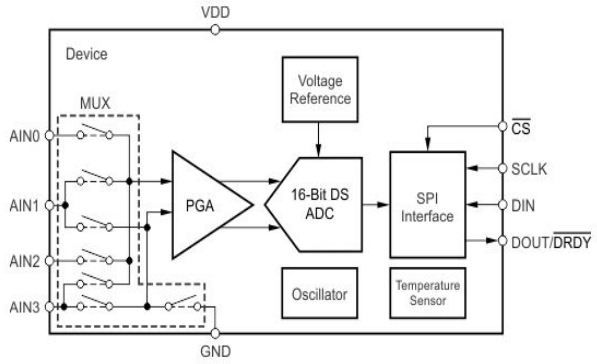


Figura 5. Diagrama funcional do ADS 1118.

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".

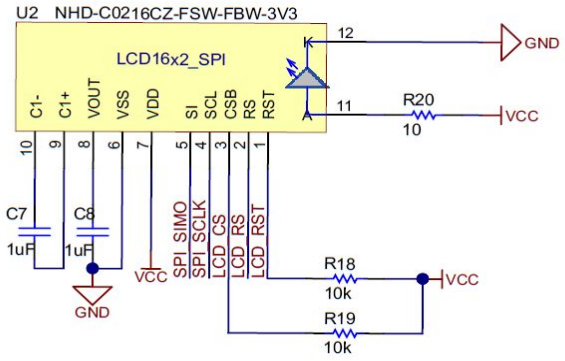


Figura 6. Esquemático do display 16x2.

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

- **Transmissão de imagens:**

Para filmar o ambiente inspecionado, a câmera escolhida, Ov5647, foi acoplada ao chassi do robô. Ela é compatível com a Raspberry Pi zero Wh, podendo ser conectada através de um cabo flat adaptado, possui Leds infravermelhos, um sensor de imagem CMOS de baixa tensão e alta performance, que fornece uma saída de vídeo com resolução máxima de 2592x1944. Dispondo de 5 megapixel - foco ajustável, fornece as imagens através do barramento de controle da câmera serial ou da interface MIPI, tem um array de imagens capaz de operar até 15 fps em resolução máxima e com controle de qualidade, transferência de dados e funções de câmera, através da interface SCCB. O núcleo do sensor gera dados de pixels contínuos a uma taxa de quadros constante, indicada por HREF e VSYNC.

A figura abaixo mostra um diagrama de blocos funcional:

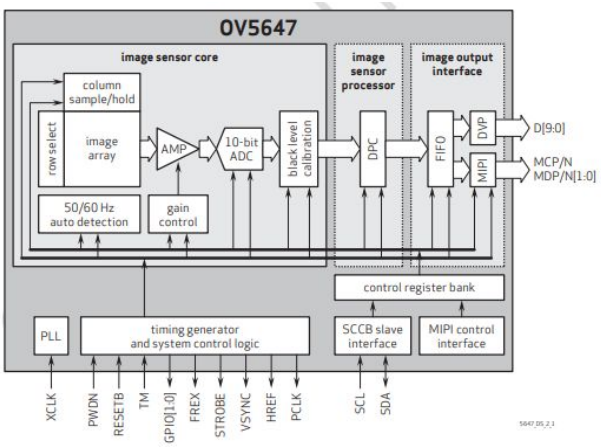


Figura 7. Diagrama de blocos da camera Ov5647.

As conexões descritas podem ser verificadas na imagem abaixo:

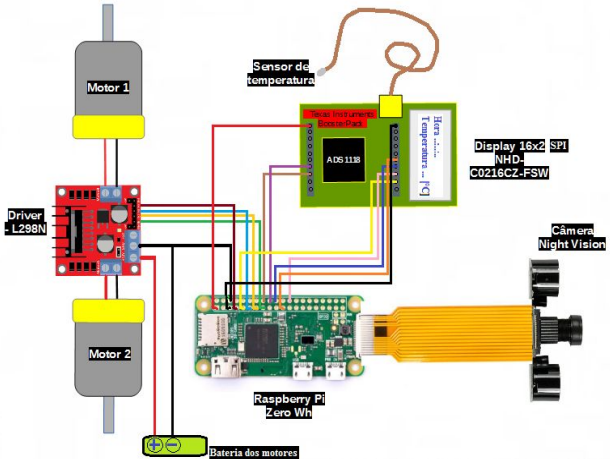


Figura 8. Esquemático de conexão unindo todos os módulos à Raspberry.

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:

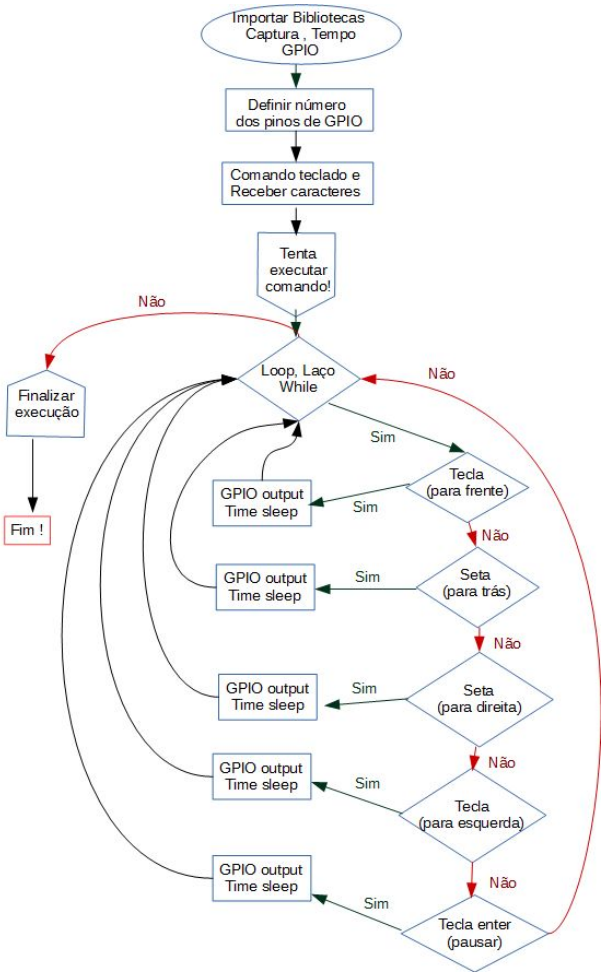


Figura 9. Fluxograma utilizado no desenvolvimento do script de controle dos motores.

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 10. Primeira parte, bibliotecas e pinos.

Foram usadas três bibliotecas, Figura 10:

➔ 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 11. Segunda parte, controle direcional.

Na Figura 11. 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 12. Terceira parte, finalização.

A terceira parte do código é a finalização e é apresentado na figura 12. 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.

O fluxograma da figura abaixo mostra um pseudocódigo com as etapas necessárias para configurar a comunicação entre o dispositivo e um microcontrolador para fazer leituras subsequentes do ADS1118. Como um exemplo, as configurações padrão do registro de configuração são alteradas para configurar o dispositivo para FSR = ± 0,512 V, conversão de modo contínuo, e uma taxa de dados de 64-SPS.

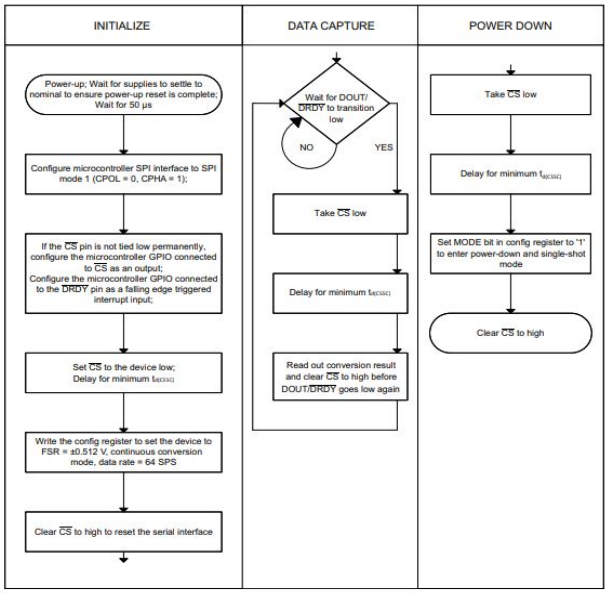


Figura 13. Fluxograma simplificado do pseudocodigo utilizado.

- **Transmissão de imagens:**

A captura e transmissão de imagens, ao vivo, foi possível através da aplicação MJPG-streamer, responsável pela leitura de arquivos JPG da câmera, compatível com Linux-UVC, e por permitir um streaming desses arquivos em formato M-JPEG, através do protocolo HTTP. [18]

Foi realizada a instalação do MJPG-streamer, com auxílio de alguns scripts de inicialização. O sistema da Raspberry Pi foi acessado via terminal SSH e o repositório abaixo foi clonado na pasta /home/pi do sistema Raspbian.

```
cd /home/pi
```

```
git clone
https://github.com/jacksonliam/mjpg
-streamer.git
```

Esse repositório é um fork do projeto original com adição de um suporte a câmera da Raspberry Pi.

Os seguintes programas, necessários para compilação do MJPG-streamer, foram instalados:

```
sudo apt-get install cmake
libjpeg8-dev
```

Em seguida, a compilação foi iniciada:

```
cd
mjpg-streamer/mjpg-streamer-experime
ntal
make
sudo make install
```

Após a compilação, obteve-se um binário chamado mjpg_streamer e também alguns plugins .so dentre eles o input_raspicam.so e output_http.so:

Os comandos abaixo foram utilizados para executar o MJPG-streamer:

```
export LD LIBRARY PATH=.
./mjpg streamer  -o  "output http.so
-w ./www" -i "input raspicam.so"
```

Logo, obteve-se os seguintes dados:

```
MJPG Streamer Version.: 2.0
i: fps.............: 5
i: resolution........: 640 x 480
i: camera parameters..............:

Sharpness 0, Contrast 0, Brightness
50
Saturation    0,    ISO    0,    Video
Stabilisation      No,      Exposure
compensation 0
Exposure   Mode   'auto',   AWB   Mode
'auto', Image Effect 'none'
Metering   Mode   'average',   Colour
Effect Enabled No with U = 128, V =
128
Rotation 0, hflip No, vflip No
ROI   x   0.000000,   y   0.000000,   w
1.000000 h 1.000000
o: www-folder-path......: ./www/
o: HTTP TCP port........: 8080
o: HTTP Listen Address..: (null)
o: username:password....: disabled
o: commands.............: enabled
i: Starting Camera
Encoder Buffer Size 81920
```

Feito isso, restou apenas verificar a filmagem do local explorado, ao vivo, na página "MJPG-streamer" em um navegador web, acessando com o IP da Raspberry na porta 8080.

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 14 apresenta os resultados obtidos impressos no terminal de programação enquanto que a figura 13 apresenta o resultado exibido no display de LCD acoplado.

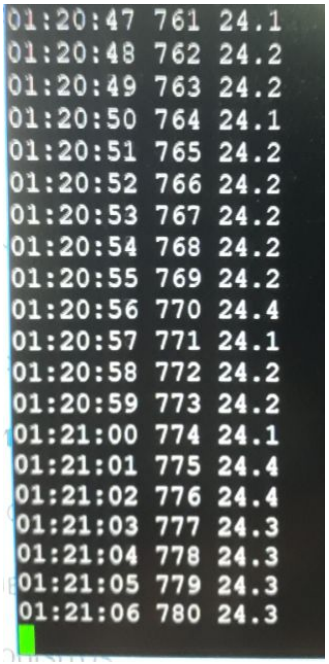


Figura 14. Resultados obtidos a cada segundo pelo sensor de temperatura.

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 15. Resultado da medida de temperatura no horário exibido.

Observar que na figura 15 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.

A terceira parte do projeto consistiu na implementação e análise do funcionamento da câmera através da aplicação MJPG-streamer .

Ao abrir um navegador e acessar o IP da Raspberry Pi na porta 8080, apareceu a página de boas vindas do MJPG-streamer.

Clicando-se na aba “Stream” o vídeo pode ser assistido em tempo real.

Como módulo desta câmera dispõe de leds infravermelhos - visão noturna, verificou-se a possibilidade de operação em ambientes com pouca luminosidade. Isso pode ser observado nas imagens a seguir:

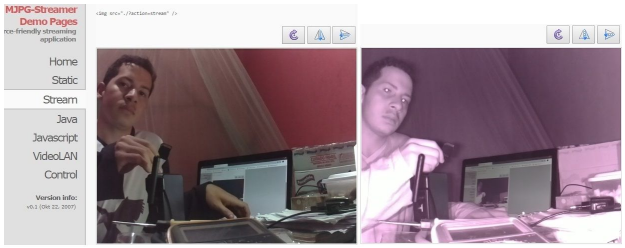


Figura 16. Imagens capturadas pela câmera acoplada ao robô, em ambientes com e sem boa iluminação.

Na figura 17 pode-se verificar o projeto implementado ainda de forma parcial:

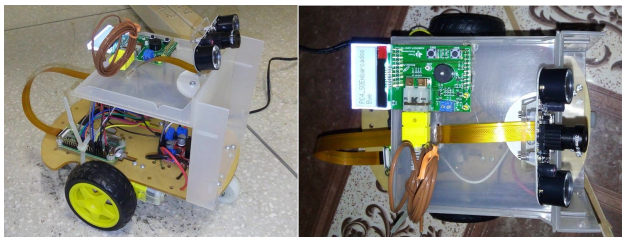


Figura 17. Robô explorador implementado até o momento.

De modo geral, o robô tem o intuito de explorar locais em que nem sempre ficará exposto somente à temperatura ambiente. Então tornou-se necessário verificar alguns parâmetros de funcionamento da placa micro controlada enquanto ela está em fase ociosa, isto é, em modo “espera”. Alguns comandos foram utilizados para obtenção destes dados, como temperatura e frequências de funcionamento da CPU. Isso é de interesse para o projeto pois quando o robô estiver em funcionamento a placa raspberry não pode exceder especificações de fábrica e algumas limitações – como temperatura de funcionamento por exemplo.

Em modo ocioso utilizamos alguns comandos para obter estes parâmetros de funcionamento. A lista abaixo relaciona cada um deles e logo em seguida um breve comentário.

- · Informações gerais da RaspBerry PI;
- · Memória da CPU x GPU;
- · Temperatura interna da placa;
- · Frequência da CPU;
- · Tensões de alguns componentes;
- · Codecs;

- · Frequência de outros componentes.

A Figura 18 apresenta as informações gerais da placa obtido através do comando:

cat /proc/cpuinfo

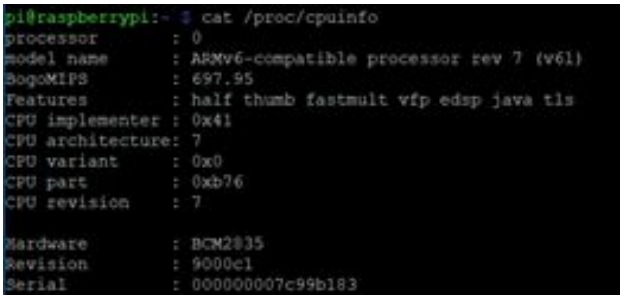


Figura 18

O comando utilizado apresenta o resultado de um diretório virtual mantido no kernel do sistema. O processador da raspberry é o BCM2835 (Broadcom) baseado na arquitetura ARMv6.

Em seguida, a Figura 19 é obtida através do comando:

vcgencmd get_mem arm && vcgencmd get_mem gpu

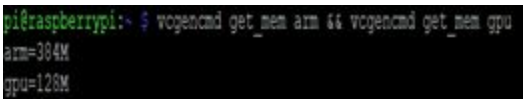


Figura 19

Pode-se ver que a memória é compartilhada entre a CPU com 384 Mb e a GPU com 128Mb, totalizando 512 Mb de memória RAM.

Para a temperatura interna atual da placa os testes foram feitos sem stress da CPU, isto é, a placa praticamente estava ligada rodando os códigos compilados para funcionamento correto do robô. De modo geral, a temperatura apresentada na Figura 20 é ambiente e normal para um componente eletrônico em “espera”. O comando utilizado foi:

vcgencmd measure_temp

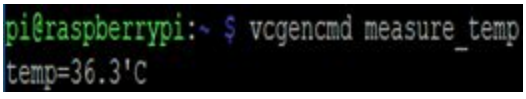


Figura 20

A Figura 21 apresenta o resultado obtido pelo comando:

cat
/sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq

Figura 21

Na figura é possível ver 3 números que representam as frequências atuais (cur), mínima (min) e max (máxima) permitidas pela CPU da placa. Observar que a temperatura da placa está intimamente ligada à frequência de operação da CPU e portanto, no momento da medição era a mínima possível praticada.

As tensões de núcleo da CPU e da memória SDRAM estão apresentadas na Figura 22. Tais tensões estão relacionadas diretamente com o clock praticado pelo núcleo da CPU e afetam também a temperatura interna. Tais dados foram obtidos pelos comandos:

for id in core sdram_c sdram_i sdram_p ; do \
echo -e “$id:\t$(vcgencmd measure_volts $id)” ; \
done

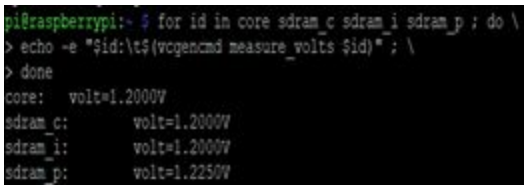


Figura 22

Os codecs ativos na placa durante a obtenção destes parâmetros estão apresentados na Figura 23 e foram obtidos com os comandos:

for codec in H264 MPG2 WVC1 MPG4 MJPG WMV9 ;
do \
echo -e "$codec:\t$(vcgencmd codec_enabled $codec)" ;
\ done

Figura 23

Completando a lista obtém-se as frequências de clock praticadas por outros componentes da placa raspberry como h264, isp, v3d, uart, pwm, emmc, pixel, vec, hdmi e dpi. A Figura 24 apresenta estes resultados, alguns deles tiveram resultado de clock zero o que indica a não utilização daquele componente no momento da medição. Isso não quer dizer que não venham a ser utilizados neste projeto no futuro. O comando utilizado para obter estes parâmetros foi:

```
for src in arm core h264 isp v3d uart pwm emmc pixel vec hdmi dpi ; do \
echo -e "$src:\t$(vcgencmd measure_clock $src)" ; \
done
```

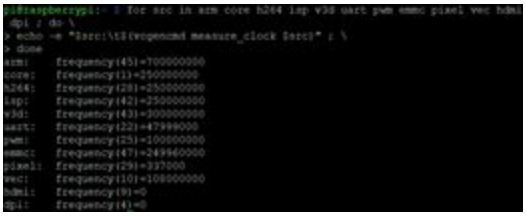


Figura 24

X. CONCLUSÃO

O protótipo do robô explorador está quase pronto, até o presente momento, as partes do controle dos motores, de monitoramento de temperatura e de transmissão de imagens em tempo real foram implementadas com sucesso.

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

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 possível fazer streaming de vídeo com o RPi Cam Web interface, usando MJPG-streamer, onde visualizou-se o vídeo em tempo real do ambiente inspecionado, com um delay inferior à 5 segundos, através de um navegador web.

Como a streaming (transmissão ao vivo), requer bastante processamento da RPi, fazendo uso do poder de processamento da CPU da Raspberry Pi, o sistema foi simplificado, dispondo de 3 threads (fluxos de execução) principais, sendo o primeiro o controle dos motores, o segundo para medir temperatura e o terceiro para transmissão de imagens em tempo real.

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, Octuber 11, 2013.

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

[17] Datasheet. Preliminary specification OV5647. Disponível em : <https://cdn.sparkfun.com/datasheets/Dev/RaspberryPi/ov5647_full.pdf > Acessado em: 6 de junho de 2019.

[18] FilipeFlop. Streaming de vídeo com Raspberry Pi Zero W. Disponível em : <https://www.filipeflop.com/blog/streaming-de-video-raspberry-pi-zero-w/> Acessado em: 7 de junho de 2019.

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()
```