Relatório do Projeto Final: Sistema de Bengala Inteligente para Auxílio a Deficientes Visuais

Yan Tavares (202014323)
LAB-SISMIC

18 de fevereiro de 2025

Sumário

1 Introdução

Este relatório apresenta o desenvolvimento e a implementação de um sistema de bengala inteligente, cujo objetivo é auxiliar a comunidade de deficientes visuais. O sistema detecta a presença de obstáculos próximos por meio do sensor de distância laser **VL53L0X** e fornece um alerta tátil através de um motor vibracall, cuja intensidade de vibração é controlada via *PWM* gerado pelo microcontrolador MSP430. Para que o usuário possa ajustar a sensibilidade desse alerta, utiliza-se um potenciômetro linear conectado ao conversor *ADC* do MSP430.

2 Metodologia

Nesta seção, descrevemos detalhadamente cada componente e como eles foram integrados no projeto, além de explicar o funcionamento do código-fonte em C, que faz uso da **API oficial da STMicroelectronics** para o sensor VL53L0X.

2.1 Componentes Utilizados e Conexões

2.1.1 Microcontrolador MSP430F5529

- **Função:** O MSP430F5529 é o microcontrolador responsável por processar as leituras dos sensores, gerar sinais PWM para acionar o motor vibracall e gerenciar a comunicação I2C com o sensor VL53L0X. Ele também realiza conversões analógicas via ADC para ler o valor do potenciômetro, permitindo o ajuste da intensidade de vibração.
- **Características:** Este microcontrolador possui uma arquitetura de baixo consumo e alta eficiência, com múltiplos periféricos integrados, incluindo temporizadores, módulos ADC e interfaces de comunicação (I2C, SPI, UART).

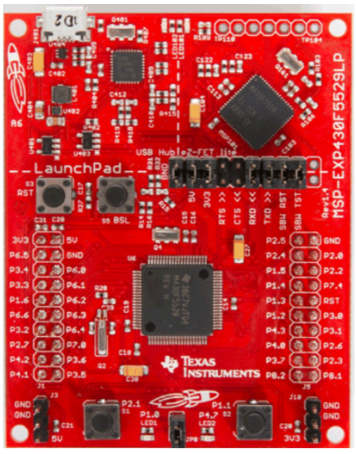

Figura 1: Microcontrolador MSP430F5529. Fonte: f5529-launchpad https://www.ti.com/.

2.1.2 Transistor BC547

- **Função:** O transistor BC547 (NPN) é utilizado como chave para acionar o motor vibracall. O sinal PWM proveniente do MSP430 (pino `P3.6`) é conectado à base do transistor, o emissor do transistor vai para o GND e o coletor é ligado ao terminal negativo do vibracall.
- **Ligação do Vibracall:** O vibracall é alimentado em +VCC (3.3 V), e o outro terminal segue para o coletor do BC547. Assim, quando o PWM está em nível alto, o transistor satura e energiza o motor, gerando a vibração.

Figura 2: Transistor BC547 (NPN). Fonte: https://huinfinito.com.br/.

2.1.3 Motor Vibracall (com pêndulo de vibração)

- **Função:** Responsável por fornecer o alerta tátil ao usuário. É acionado pelo transistor BC547.
- **Alimentação:** Conectado em +VCC (pino de alimentação do sistema) e, do outro lado, ao coletor do BC547.

Figura 3: Motor vibracall 4x10 mm. Fonte: https://huinfinito.com.br/.

2.1.4 Potenciômetro Linear

- **Função:** Ajustar a intensidade máxima de vibração conforme a preferência do usuário.

- **Ligação:** Terminal central do potenciômetro ligado ao pino `P6.1` (ADC do MSP430). Um dos terminais externos ligado em +VCC e o outro terminal externo ligado em GND.

Figura 4: Potenciômetro linear (10 kΩ). Fonte: https://huinfinito.com.br/.

2.1.5 Sensor VL53L0X

- **Função:** Medir a distância até objetos próximos por meio de um feixe de luz infravermelha.
- **Comunicação I2C:** Conectado ao MSP430 nos pinos `P3.0` (SDA) e `P3.1` (SCL). A alimentação do sensor deve ser compatível (3.3 V na maioria dos módulos).
- **API STMicroelectronics:** O código faz uso de `vl53l0x_api.h` e `vl53l0x_platform.h` para inicialização e leitura do sensor.
- **Características:** O VL53L0X opera emitindo um feixe de laser infravermelho pulsado a partir de um emissor de diodo laser embutido. Esse feixe é refletido pelo objeto e retorna ao sensor, onde é detectado por um fotodiodo sensível à luz infravermelha.

Figura 5: Sensor VL53L0X. Fonte: https://easytronics.com.br/.

2.2 Esquema de Conexões

A Figura 6 mostra um diagrama de como todos os componentes são conectados ao MSP430:

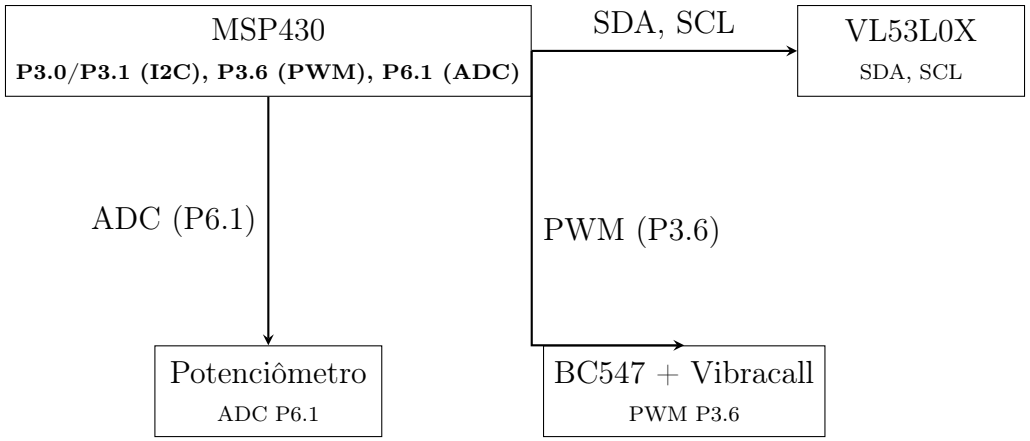

Figura 6: Diagrama de conexões entre MSP430, VL53L0X, potenciômetro e transistor + vibracall.

2.3 Configuração de Periféricos e Explicação do Código

O código-fonte em C para o MSP430 está organizado de forma a inicializar periféricos e executar o *loop* principal de detecção de distância e acionamento do vibracall. A seguir, descrevemos cada parte:

2.3.1 Função `main()`

- **Desativar WDT:**

  ```
  WDTCTL = WDTPW | WDTHOLD;
  ```

  Impede o reset automático do watchdog durante o desenvolvimento.

- **Configurações iniciais:**
  - `configPWM()`: Configura o Timer B0 para gerar PWM no pino `P3.6` (TB0.6).
  - `configI2C()`: Configura a interface I2C nos pinos `P3.0` (SDA) e `P3.1` (SCL).
- **Inicialização do Sensor VL53L0X (Funções da API oficial da STMicroelectronics):**
  - `VL53L0X_DataInit(&dev)`: Inicializa a estrutura do dispositivo.
  - `VL53L0X_StaticInit(&dev)`: Configura parâmetros estáticos do sensor.
  - `VL53L0X_PerformRefCalibration(&dev, &VhvSettings, &PhaseCal)`: Executa calibração interna.
  - `VL53L0X_PerformRefSpadManagement(&dev, &refSpadCount, &isApertureSpads)`: Gerencia SPADs de referência.
  - `VL53L0X_SetDeviceMode(&dev, VL53L0X_DEVICEMODE_SINGLE_RANGING)`: Modo de medição única.

- `VL53L0X_SetLimitCheckEnable(...)`: Habilita verificações de limite (*sigma*, *signal rate*, etc.).

- **Loop Principal:**

  1. `ADCRead(POTENTIOMETER_PIN)` lê o valor do potenciômetro no pino `P6.1` e atribui a `potValue`.
  2. Ajusta-se o valor de `potValue` para no máximo 4000, criando um limite de leitura.
  3. `VL53L0X_PerformSingleRangingMeasurement(&dev, &RangingMeasurementData)` obtém a distância em milímetros.
  4. De acordo com `distMili`, é calculado o `duty` para o PWM:
     - Se `distMili` $>$ 300 mm, então `duty = 0` (sem vibração).
     - Se `distMili` $<$ 50 mm, então `duty = max_duty`.
     - Caso contrário, o `duty` varia linearmente entre 0 e `max_duty`.
  5. `TB0CCR6 = duty` ajusta o ciclo ativo do PWM no pino `P3.6`, acionando o transistor BC547, que por sua vez energiza o vibracall.

2.3.2 Função `configPWM()`

```
static void configPWM(void)
{
    P3DIR |= BIT6;
    P3SEL |= BIT6;

    TB0CTL = TBSSEL__SMCLK | MC__UP | TBCLR;
    TB0CCR0 = 10000;
    TB0CCTL6 = OUTMOD_7;
    TB0CCR6 = 0;
}
```

- Configura `P3.6` como saída de PWM (`P3DIR` e `P3SEL`).
- Utiliza o *Timer B0* no modo `UP` com clock SMCLK.
- `TB0CCR0 = 10000` define o período do PWM.
- `TB0CCR6` controla o *duty cycle* (inicialmente zero).

2.3.3 Função `configI2C()`

```
static void configI2C(void)
{
    UCB0CTL1 |= UCSWRST;
    UCB0CTL0 = UCMST | UCMODE_3 | UCSYNC;
    UCB0CTL1 = UCSSEL__SMCLK | UCSWRST;
    UCB0BRW = 10;
```

```
    P3SEL |= BIT0 | BIT1;
    UCB0CTL1 &= ~UCSWRST;
}
```

- Habilita o *software reset* (`UCSWRST`) e configura o módulo `UCB0` como mestre I2C (`UCMST`, `UCMODE_3`).
- Seta a taxa de *baud* para 10 (`UCB0BRW = 10`), que depende do SMCLK configurado.
- Seleciona os pinos `P3.0` e `P3.1` para SDA e SCL, respectivamente.
- Desativa o *software reset* para iniciar a comunicação I2C.

2.3.4 Função `ADCRead()`

```
uint16_t ADCRead(uint8_t pin)
{
    ADC12CTL0 &= ~ADC12ENC;
    ADC12CTL0 = ADC12SHT0_3 | ADC12ON;
    ADC12CTL1 = ADC12SHP | ADC12CONSEQ_0;
    ADC12CTL2 = ADC12RES_2;

    P6SEL |= (1 << pin);
    ADC12MCTL0 = pin;

    ADC12CTL0 |= ADC12ENC;
    ADC12CTL0 |= ADC12SC;

    while (!(ADC12IFG & BIT0));

    return ADC12MEM0;
}
```

- Desabilita a conversão (`ADC12ENC`), configura o ADC em 12 bits (`ADC12RES_2`) e seleciona o pino analógico (`P6SEL`).
- Inicia a conversão (`ADC12SC`) e aguarda o *flag* de conclusão.
- Retorna o valor convertido do `ADC12MEM0`.

2.4 Implementação das Funções de Escrita e Leitura I2C

Nesta seção, apresentamos as funções `VL53L0X_write_multi` e `VL53L0X_read_multi`, desenvolvidas para o microcontrolador **MSP430F5529** utilizando o módulo I2C `UCB0`. Essas funções fazem parte da API oficial da STMicroelectronics para o sensor VL53L0X e são responsáveis por realizar operações de escrita e leitura de múltiplos bytes através do barramento I2C. É importante ressaltar que, embora a API oficial forneça a estrutura e a lógica para a comunicação com o sensor, estas funções devem ser implementadas de forma específica para cada microcontrolador, de acordo com seus registradores e modos

de operação. No caso do MSP430F5529, a implementação utiliza o módulo I2C `UCB0` e os respectivos registradores para gerenciar a transmissão e recepção de dados.

2.4.1 `VL53L0X_write_multi`

A função `VL53L0X_write_multi` escreve um conjunto de bytes, armazenados em `pdata` com tamanho `count`, no dispositivo com endereço `address`, a partir do índice (registro) `index`. O fluxo de execução desta função é o seguinte:

1. Configura o registrador `UCB0I2CSA` com o endereço do dispositivo.
2. Ativa o modo transmissor e gera uma condição de `START` ao definir os bits `UCTR` e `UCTXSTT`.
3. Aguarda a disponibilidade do buffer de transmissão (flag `UCTXIFG`).
4. Envia o índice (registro) onde os dados serão escritos.
5. Aguarda a finalização da condição de `START`.
6. Verifica se ocorreu um `NACK` (não reconhecimento) e, se sim, emite uma condição de `STOP` e retorna `STATUS_FAIL`.
7. Envia, em um laço, cada byte de `pdata` após aguardar a disponibilidade do buffer.
8. Após o envio de todos os bytes, aguarda a conclusão do envio, emite uma condição de `STOP` e espera que a operação seja finalizada.
9. Retorna `STATUS_OK` se a operação for concluída com sucesso.

O código da função é apresentado a seguir:

```
int32_t VL53L0X_write_multi(uint8_t address,
                            uint8_t index, uint8_t *pdata, int32_t count)
{
    int32_t i;
    UCB0I2CSA = address;
    UCB0CTL1 |= UCTR | UCTXSTT;
    while (!(UCB0IFG & UCTXIFG));
    UCB0TXBUF = index;
    while (UCB0CTL1 & UCTXSTT);
    if (UCB0IFG & UCNACKIFG)
    {
        UCB0CTL1 |= UCTXSTP;
        while (UCB0CTL1 & UCTXSTP);
        return STATUS_FAIL;
    }
    for (i = 0; i < count; i++)
    {
        while (!(UCB0IFG & UCTXIFG));
```

```
        UCB0TXBUF = pdata[i];
    }
    while (!(UCB0IFG & UCTXIFG));
    UCB0CTL1 |= UCTXSTP;
    while (UCB0CTL1 & UCTXSTP);
    return STATUS_OK;
}
```

2.4.2 VL53L0X_read_multi

A função `VL53L0X_read_multi` realiza a leitura de `count` bytes do dispositivo com endereço `address`, a partir do índice `index`, armazenando os dados lidos em `pdata`. O fluxo de execução desta função é o seguinte:

1. Configura o registrador `UCB0I2CSA` com o endereço do dispositivo.
2. Ativa o modo transmissor e gera uma condição de `START` para enviar o índice do registro desejado.
3. Aguarda a disponibilidade do buffer de transmissão (flag `UCTXIFG`).
4. Envia o índice (registro) que se deseja ler.
5. Aguarda a finalização da condição de `START`.
6. Altera o modo para receptor, limpando o bit `UCTR`, e gera uma nova condição de `START` para iniciar a leitura.
7. Aguarda a finalização do segundo `START`.
8. Se a leitura for de um único byte, emite imediatamente a condição de `STOP`, aguarda a recepção do byte e o armazena.
9. Se a leitura for de múltiplos bytes, entra em um laço para ler `count - 1` bytes, emitindo a condição de `STOP` no penúltimo byte, e depois lê o último byte.
10. Aguarda a finalização da condição de `STOP` e retorna `STATUS_OK` para indicar o sucesso da operação.

O código da função é apresentado a seguir:

```
int32_t VL53L0X_read_multi(uint8_t address, uint8_t index,
                           uint8_t *pdata, int32_t count)
{
    int32_t i;
    UCB0I2CSA = address;
    UCB0CTL1 |= UCTR | UCTXSTT;
    while (!(UCB0IFG & UCTXIFG));
    UCB0TXBUF = index;
    while (UCB0CTL1 & UCTXSTT);
```

```
    UCB0CTL1 &= ~UCTR;
    UCB0CTL1 |= UCTXSTT;
    while (UCB0CTL1 & UCTXSTT);
    if (count == 1)
    {
        UCB0CTL1 |= UCTXSTP;
        while (!(UCB0IFG & UCRXIFG));
        pdata[0] = UCB0RXBUF;
    }
    else
    {
        for (i = 0; i < count - 1; i++)
        {
            while (!(UCB0IFG & UCRXIFG));
            pdata[i] = UCB0RXBUF;
            if (i == count - 2)
            {
                UCB0CTL1 |= UCTXSTP;
            }
        }
        while (!(UCB0IFG & UCRXIFG));
        pdata[count - 1] = UCB0RXBUF;
    }
    while (UCB0CTL1 & UCTXSTP);
    return STATUS_OK;
}
```

3 Resultados e Conclusões

Com a configuração e o código em execução, o sistema detecta objetos próximos utilizando o sensor VL53L0X. O potenciômetro, conectado ao ADC no pino `P6.1`, permite ajustar o valor máximo do duty cycle, possibilitando a personalização do feedback vibratório.

Os testes com o protótipo indicaram que o sistema responde às variações de distância, fornecendo um alerta tátil ajustável pelo usuário. Os parâmetros de intensidade de vibração e o controle via potenciômetro serão refinados com base em testes práticos e no feedback dos usuários durante a fase de prototipagem.

Testes iniciais foram realizados em distâncias curtas. Experimentos com o sensor VL53L0X demonstraram que é possível detectar objetos a distâncias maiores, o que poderá ser explorado em futuras versões do sistema.

3.1 Trabalhos Futuros

- **Montagem física na bengala:** Integrar o sensor, motor, MCU e bateria em um suporte fixado na bengala.
- **Tratamento de erros:** Implementar rotinas de *timeout* e *retry* na comunicação

I2C para reduzir falhas e bloqueios.

- **Agregação de mais sensores:** Expandir o sistema para utilizar três sensores de distância (direita, esquerda e central).
- **Testes de campo:** Avaliar a usabilidade do sistema com usuários reais em diferentes ambientes.

4 Referências

- **STMicroelectronics - API VL53L0X:** https://www.st.com/en/imaging-and-photonics-solutions/vl53l0x.html (acessado em 18 de fevereiro de 2025).
- **Texas Instruments - MSP430F5529:** http://www.ti.com/product/MSP430F5529 (acessado em 18 de fevereiro de 2025).
- **HUINFINITO:** https://huinfinito.com.br/.