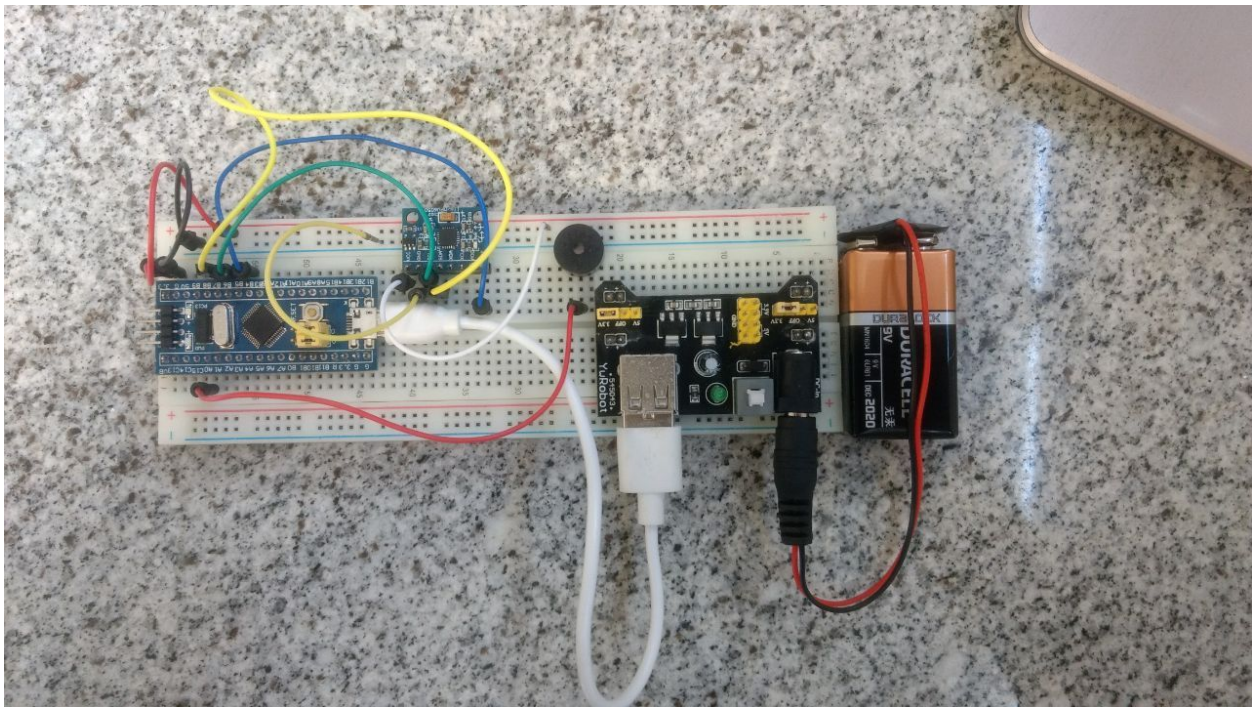


Pulseira detectora de quedas para idosos

15/12/2017

Felipe Araújo - 378599

Ismael José de Souza - 385196

Mateus Melo - 385209

Sistemas Microprocessados - Prof. Ricardo Jardel Siqueira

DETI - UFC

Introdução

A detecção de quedas é um grande desafio no domínio da saúde pública, especialmente para os idosos. Para minimizar os efeitos de quedas neste contexto, desenvolvemos um dispositivo portátil utilizando uma Bluepill STM32F103C8 e um MPU-6050.

O sistema monitora os movimentos do corpo humano, reconhece uma queda das atividades diárias normais por um algoritmo desenvolvido pelos estudantes que elaboraram este documento.

As quedas dos idosos sempre levam a graves problemas de saúde como o declínio da sua aptidão física. A fratura é a lesão mais comum na queda de um idoso e também há uma certa possibilidade de coma, trauma cerebral e paralisia. Na maioria das situações de queda, o processo de queda é a principal fonte de lesões devido ao alto impacto. Mas às vezes o resgate médico atrasado pode piorar a situação. Isso significa que, quanto mais rápido venha o resgate, menos risco enfrentarão os idosos.

O progresso da tecnologia traz mais possibilidades para nos ajudar a proteger os idosos. Os componentes de baixo consumo de energia possibilitam a realização de um dispositivo de monitoramento portátil.

Vários tipos de métodos de detecção de queda foram desenvolvidos ou aplicados em nossa vida. Um deles é o método baseado em visão por computador. As câmeras são distribuídas em espaço limitado para oferecer imagens ou vídeos de atividades humanas para implementar o algoritmo de detecção de queda. Suportes externos como sensores de movimento podem ser usados para aprimorar o método de detecção de queda baseado em visão computacional, e um algoritmo de fusão de dados pode operar a validação e correlação entre os dois subsistemas para aumentar o desempenho robusto da detecção de queda. Esses métodos baseados em computador funcionam efetivamente no ambiente interno, mas são difíceis de perceber no ambiente externo, pois a implantação de câmeras é sempre limitada.

O método baseado em sensor de movimento também é comumente usado. O acelerômetro e o giroscópio podem fornecer informações de movimento linear e angular diretamente. As medições do sensor ou a sua adequada fusão podem ser usadas para distinguir uma queda real. Existem vários tipos de métodos de detecção que diferem na constituição de sensores de movimento e algoritmos de detecção. O primeiro tipo de método de detecção é o uso de um acelerômetro. Um único acelerômetro triaxial pode fornecer acelerações de objetos em três direções que incluem a influência da gravidade.

Uma coordenada será construída quando o acelerômetro for fixado no corpo humano. A influência da gravidade ou aceleração dinâmica está disponível usando um filtro de passagem baixa ou um filtro de passagem alta. Alguns tipos de informações de movimento angular também podem ser calculados com base na relação entre componentes de aceleração e sua soma vetorial. O segundo tipo de método de detecção é baseado em acelerômetro e giroscópio. O giroscópio pode oferecer velocidade angular e o acelerômetro pode oferecer informações de movimento linear. O terceiro tipo de método de detecção também usa um magnetômetro. Um magnetômetro triaxial pode detectar a força magnética em três direções, e também pode fornecer informações de movimento angular no plano horizontal. Mas o campo magnético do ambiente pode perturbar o campo geomagnético, o que reduz a confiabilidade das saídas do magnetômetro, por exemplo, em alguma arquitetura de estrutura de aço ou perto de alguns objetos com forte eletromagnetismo. Como informações angulares também podem ser extraídas das medidas do acelerômetro, um filtro de espaço de estado, como o filtro de Kalman, é uma técnica comumente usada para combinar informações de movimento angular. Além disso, sensores como o barômetro também podem auxiliar sensores de movimento puro no reconhecimento da marcha humana.

Mas, de fato, usar mais sensores significa mais consumo de energia, e é um desafio criar um algoritmo apropriado para fundir diferentes tipos de sensores. Um único acelerômetro triaxial é suficiente para a detecção de queda humana, pois informações suficientes podem ser extraídas de suas medidas. Além disso, a coordenada do acelerômetro não precisa ser corrigida se apenas a magnitude do vetor de soma for necessária, e isso é bastante conveniente para a aplicação wearable. Neste documento, é desenvolvido um sistema de detecção de queda baseado em um dispositivo portátil(STM32F103C8). O dispositivo é programado com um algoritmo de detecção de queda eficiente com menos consumo de energia e recursos, o que significa que é um design apropriado para aplicações ao ar livre.

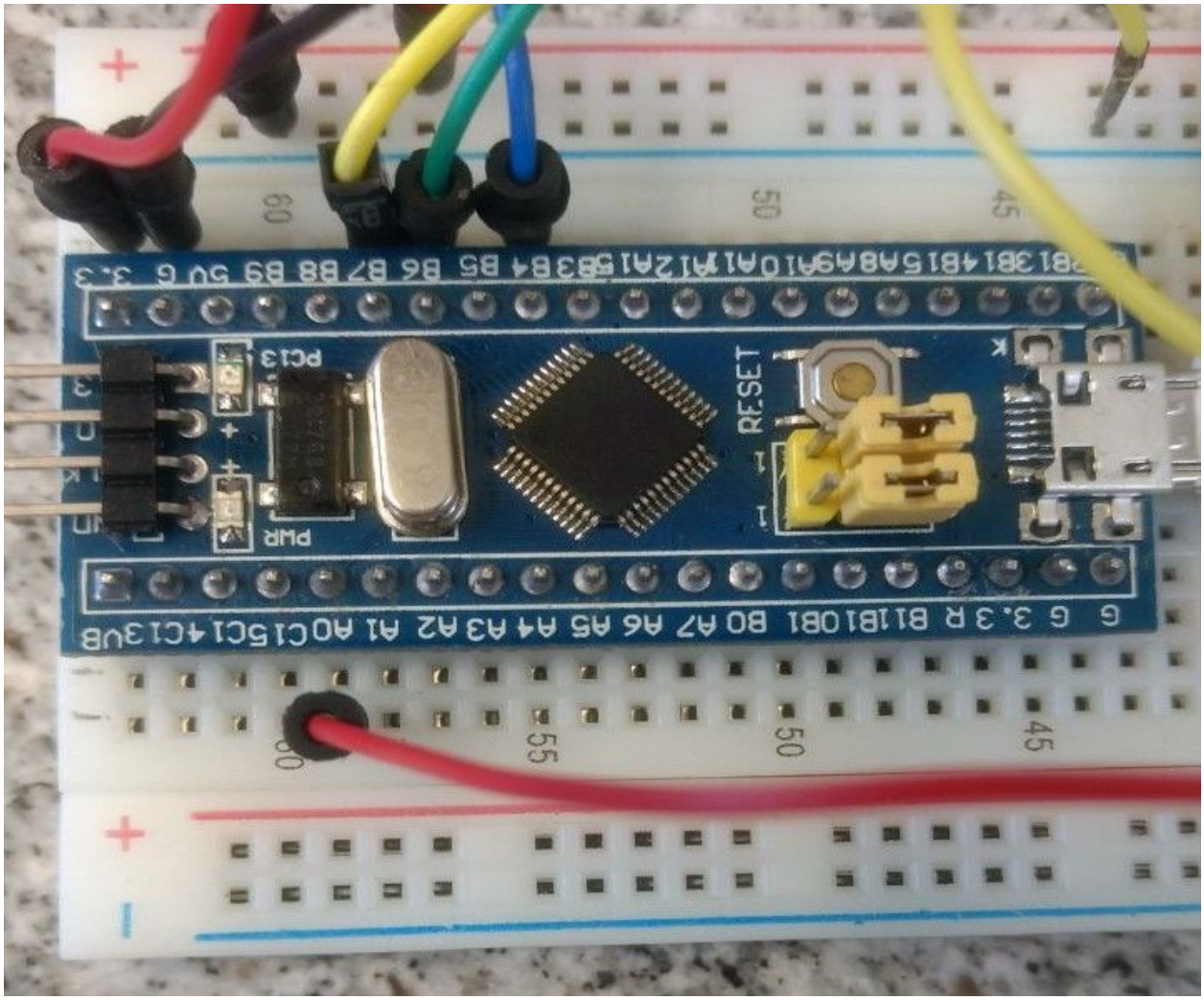
Goals

1. Detectar quedas a partir do acelerômetro MPU-6050.
2. Acionar o alarme (no caso, buzzer) como um alerta.

Especificações

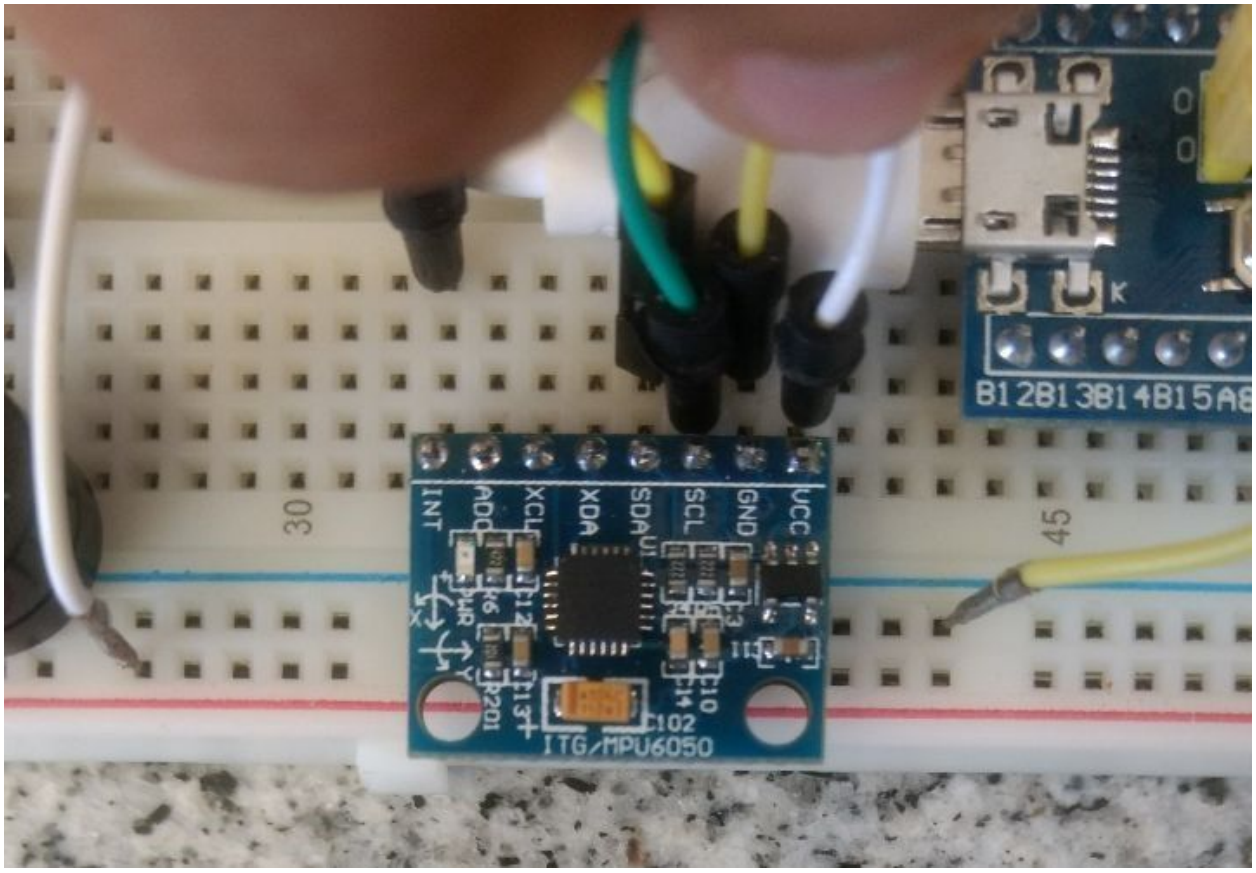
Os componentes utilizados no projeto foram:

1. Microprocessador STM32F103C8.



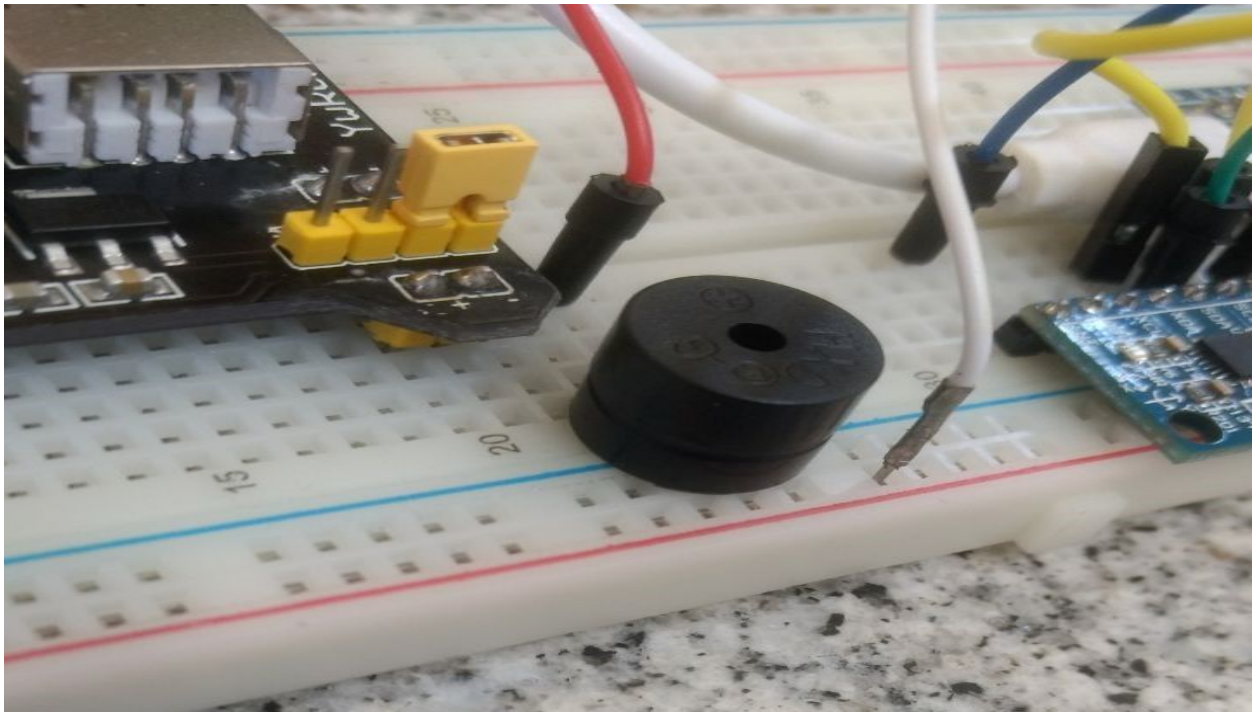
O microprocessador STM32F103C8 é um microprocessador que tem clock de 72Mhz, 40 pinos com suas respectivos tipos e funções, possui um botão de reset de fácil acesso na placa, comunicação via USB, I2C, SPI e possui pinagem para que seja utilizado o programador da STlink.

2. Acelerômetro.



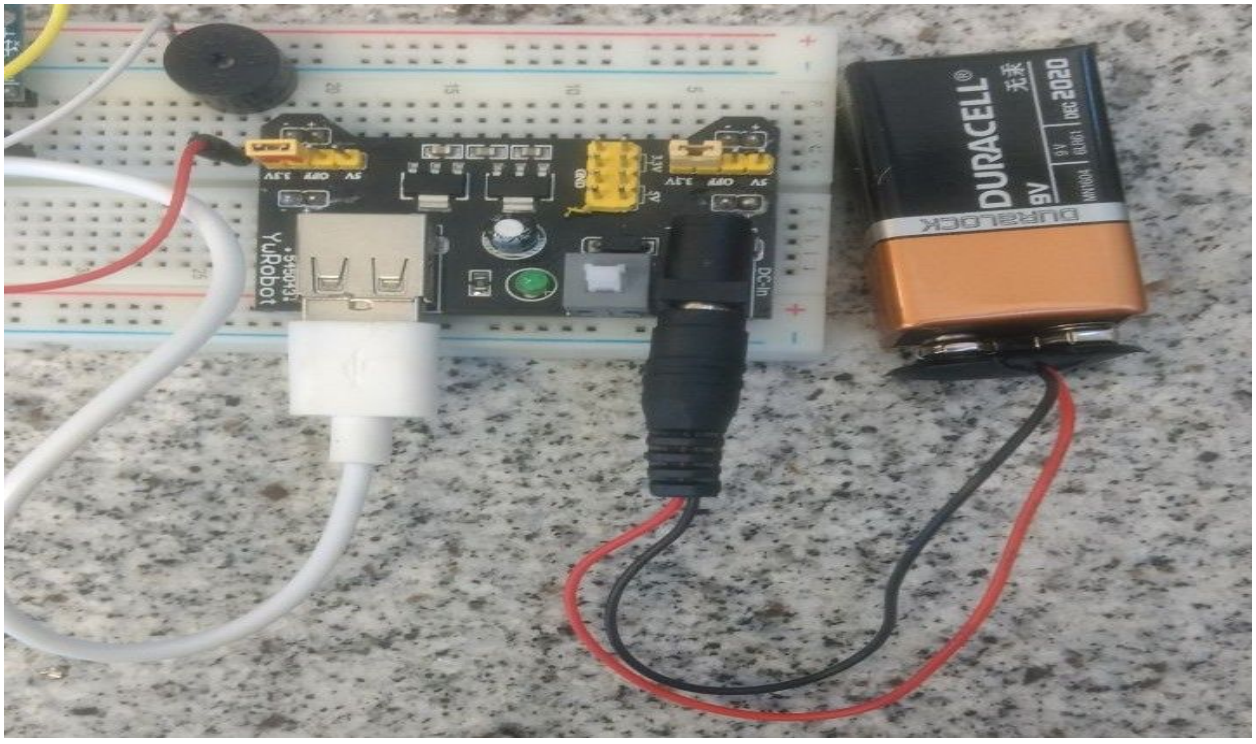
O dispositivo MPU-6050, usado no projeto com a função de acelerômetro, também possui a de giroscópio e magnetômetro, a sua vantagem é o fato de possuir uma vasta quantidade de materiais na internet e experimentos utilizando-o. A comunicação desse dispositivo com a placa é realizado por conexão I2C, no qual abordaremos posteriormente.

3. Buzzer.



O buzzer é o dispositivo responsável pelo alerta no caso de ocorrer quedas. O seu funcionamento no circuito do projeto, é igual ao de um Led comum.

4. Fonte de alimentação externa (YwRobot - 545043).



A fonte de alimentação externa é responsável pela alimentação do circuito, tendo em vista que o dispositivo encontra-se junto ao idoso que o utiliza, necessitando de uma fonte móvel.

Esquematização

I. STM32 CubeMX

Essa ferramenta é da própria STMicroeletronics, que nos possibilita configurar previamente os pinos do microprocessador a fim de gerar automaticamente o código fonte base para a nossa aplicação.

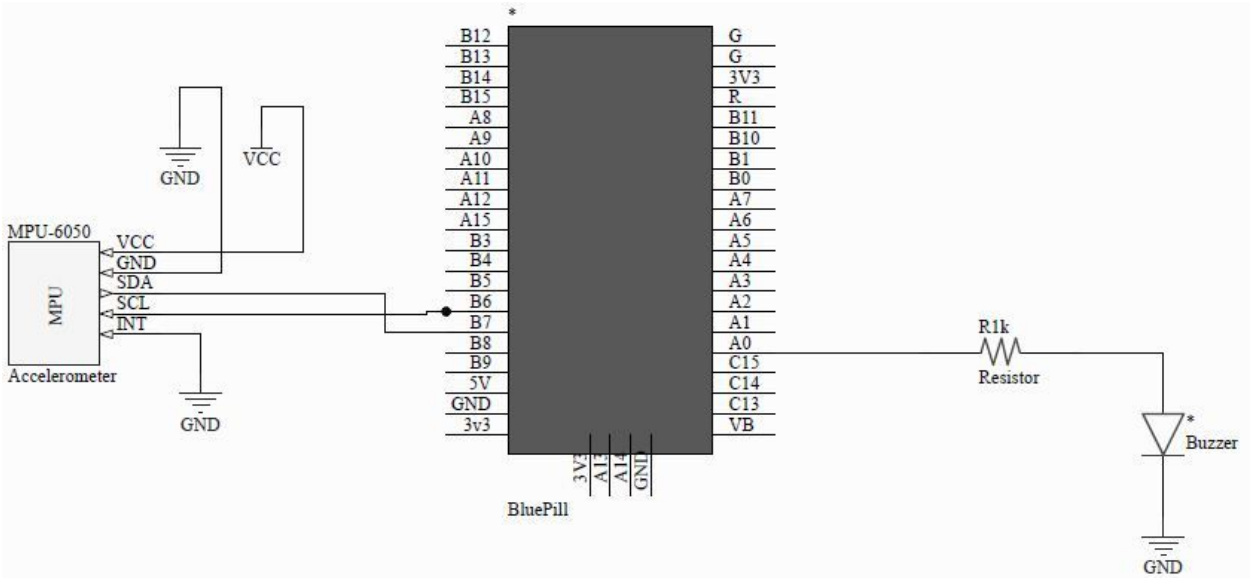
Para o projeto, utilizamos as portas PB6 e PB7 para comunicação I2C com o MPU-6050, para isso, configura-se a opção I2C1 como "I2C". Utiliza-se também a porta PA0 e PC13, onde PA0 é configurado para GPIO OUTPUT e PC13 é utilizado para apenas ligar o led da bluepill, indicando o seu funcionamento.

O pino PB6 é, por padrão, o pino de clock que se comunica com o dispositivo, gerando a taxa de frequência que será feita a troca de dados.

O pino PB7 é por onde esses dados serão lidos, vale ressaltar que na configuração I2C a troca de dados é apenas dispositivo - microcontrolador, half duplex.

II. Conexões

Os pinos devem ser devidamente conectados como se segue no esquemático:



-Pino SDA da MPU-6050 deve estar ligado no PB6 da Bluepill.
-Pino SCL da MPU-6050 deve estar ligado no PB7 da Bluepill.
-Pino INT conectado em 0v Logico.
-Pinos de alimentação conectados.
-Pino de PB0(GPIO output) da Bluepill conectado ao resistor que se conecta com o buzzer.

Algoritmo

Primeiramente, é necessária uma constante (ACCELERATION_THRESHOLD) para medir o valor máximo de diferença de aceleração, o que permite verificar uma queda no eixo z.

No tempo x, é verificado a aceleração. No tempo (x+0,5 segundo) é novamente verificado a aceleração para calcular o delta de aceleração. Uma vez que o delta seja maior que ACCELERATION_THRESHOLD, o idoso provavelmente caiu.

O segundo passo é verificar se o idoso permaneceu no chão. Isto se deve ao fato de que, se ele caiu e levantou, então não é tão urgente quanto uma queda sem após levantar-se.

Por fim, o ciclo se repete.

```
bool calculate_fall() {
   int initialAcceleration = get_norm();
   HAL_Delay(SHORT_SLEEP);
   int finalAcceleration = get_norm();

   if (finalAcceleration - initialAcceleration > ACCELERATION_THRESHOLD) {
         HAL_Delay(LONG_SLEEP*3);
         int acceleration = get_norm();
         HAL_Delay(SHORT_SLEEP);

         for (int i=0; i<10; i++) {
               int delta = get_norm() - acceleration;
               acceleration = get_norm();

               if (delta > ACCELERATION_THRESHOLD_AFTER_VARIATION) {
```

```
                    return false;
                }
                HAL_Delay(SHORT_SLEEP);
            }
            return true;
        }
        return false;
    }
```