Ponto de Controle 4

Microprocessadores e Microcontroladores

Vitor Carvalho de Oliveira
RA: 14/0165498
Universidade de Brasília, Campus Gama
vitor.carvalhoo@icloud.com

Douglas Afonso Xavier de Rezende
RA: 13/0107816
Universidade de Brasília, Campus Gama
douglasafonso@outlook.com

***Resumo* — O intuito deste projeto é modernizar um clássico jogo chamado Marble Maze, incluindo neste a eletrônica embarcada, utilizando o microcontrolador MSP430, para auxiliar na reabilitação de pessoas que sofreram algum tipo de lesão envolvendo a região do punho ou que possuem algum tipo de patologia específica desta região, além de melhorar a coordenação motora e desenvolver um maior envolvimento cognitivo.**

Palavras-chave* – *Jogo, MSP430, Reabilitação, Punho, Microcontroladores, Eletrônico, Movimento.

I. INTRODUÇÃO

Nos últimos anos, os jogos eletrônicos e digitais de movimento, assim chamados por tornarem importante o corpo e os movimentos na interação com o jogo, ganharam visibilidade e passaram a receber grandes investimentos da indústria de jogos eletrônicos. Grandes empresas como; Sony, Microsoft e Nintendo investiram fortemente em acessórios e periféricos que dessem suporte para jogos de movimento na sétima geração de consoles (Ps3, Xbox 360 e Wii).

Figura 1: Acessórios que dão suporte aos jogos de movimento da sétima geração de consoles.

Muitos dos jogos de movimento desenvolvidos, envolviam e simulavam práticas esportivas, como por exemplo, lutas, corrida, "snowboard", golf, entre outros, que estimulavam os jogadores à prática de exercícios físicos, promovendo gasto calórico e melhorando a coordenação motora dos jogadores, além da diversão e do entretenimento garantido.

Atualmente os jogos eletrônicos de movimento têm ganhado um novo propósito, o de auxiliar na reabilitação de pessoas com alguma deficiência motora ou que sofreram algum tipo de lesão corporal que necessitam de sessões de fisioterapia para recuperar certos movimentos. Segundo o terapeuta ocupacional Fábio Galvão, formado pela Universidade de Potiguar, a união dos jogos eletrônicos de movimento com o acompanhamento profissional, pode trazer uma série de benefícios, tais como:

- Fortalecimento muscular;
- Melhorar a amplitude dos movimentos;
- Ampliação da atividade cerebelar;
- Estímulo à concentração e ao equilíbrio;
- Sensação de superação;
- Maior envolvimento cognitivo;

Já existem relatos da utilização, com resultados satisfatórios, da Terapia Ocupacional e da Fisioterapia, aliadas aos jogos eletrônicos de movimento em vários países, como Estados Unidos, Alemanha, Inglaterra, entre outros.

Seguindo essa tendência, este projeto visa modernizar e incluir elementos da eletrônica embarcada em um clássico jogo chamado "Marble Maze" (Labirinto) para que ele possa ser utilizado na reabilitação de pessoas com algum tipo de deficiência motora no punho. Este jogo criado em 1946 pela empresa de brinquedos BRIO, consiste em um tabuleiro, geralmente de madeira, com um labirinto e obstáculos, cujo objetivo é levar uma bola de gude até um ponto específico do labirinto por meio de inclinações manuais do tabuleiro.

Figura 2: Jogo Marble Maze.

II. JUSTIFICATIVA

Pessoas que sofreram fratura de punho: Ossos do antebraço na região distal tanto no osso do rádio quanto no osso da ulna ou lesões de tendões: tendão flexor e extensor de punho de dedos ou que convivem com alguma patologia específica de punho: Síndrome do Túnel do Carpo, Síndrome de Canal de Guyon, Tendinites, Cistos Sinoviais, Dedo em gatilho, Artrite localizada no punho, Síndrome do Túnel Radial, Tenossinovite de Quervain, ou que foram vítimas de um Acidente Vascular Cerebral (AVC) ou paralisia cerebral que sofrem de espasticidade (distúrbio de controle muscular caracterizado por músculos tensos ou rígidos) geralmente passam por diversas sessões intensivas e massivas de fisioterapia.

Essas sessões, apesar de serem fundamentais na reabilitação e estimularem os pacientes a se manterem sempre ativos, fortalecendo músculos, articulações, e promovendo uma recuperação eficaz, na maioria dos casos, são vistas como monótonas, repetitivas e entediosas pelos pacientes.

A motivação desse projeto, portanto, é tornar essas sessões de reabilitação do punho mais atraentes, divertidas, estimulantes e lúdicas para os pacientes, mediante a utilização da versão eletrônica do jogo Marble Maze.

Os movimentos fisiológicos executados pelo punho e os graus de amplitude de movimento são: Flexão (para baixo 90º) extensão (para cima 70º) desvio radial (para o lado do polegar 20º) desvio ulnar (para o lado do dedo mínimo 45º) pronação (90º) e supinação (90º). Esses são os movimentos que são explorados na fisioterapia.

Figura 3: Exemplo de movimento fisiológico do punho.

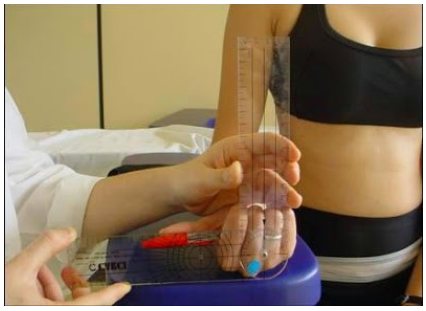

Figura 4: Exemplo de movimento fisiológico do punho.

Figura 5: Exemplo de movimento fisiológico do punho.

III. OBJETIVOS

A proposta deste projeto é modernizar e incluir a eletrônica embarcada utilizando o MSP430 em um clássico jogo chamado "Marble Maze" O objetivo é que os movimentos fisiológicos do punho sejam os responsáveis por inclinar o tabuleiro, e para isso, será utilizado sensores de movimento (giroscópio e acelerômetro) para capturar as rotações e inclinações do punho e traduzir em sinais que atuarão em servomotores que estarão distribuídos por baixo do tabuleiro para realizar as inclinações.

Com isso o jogo visa ser um auxiliador na reabilitação de pessoas com alguma deficiência motora na região do punho, promovendo a exercitação focalizada dos movimentos fisiológicos do punho e trabalhando a amplitude desses movimentos, além de proporcionar sessões de fisioterapia mais diversificadas, como citado anteriormente.

IV. REQUISITOS

O jogo deverá ser capaz de identificar os 3 eixos de inclinação/movimentação do punho além dos eixos combinados de outros dois eixos, por exemplo as inclinações nas diagonais, e para isso será utilizado um módulo eletrônico que contem giroscópio e acelerômetro, para capturar essas inclinações e traduzir em sinais:

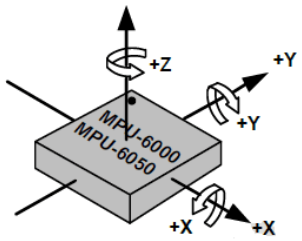

Figura 6: Módulo giroscópio e acelerômetro e os graus de liberdade possíveis.

Esses sinais serão enviados para o MSP430 que irá traduzí-los e convertê-los em sinais de saída para os servomotores, distribuídos embaixo do tabuleiro, para realizar as inclinações de acordo com o grau de amplitude do movimento do punho.

O jogador deverá segurar o módulo giroscópio/acelerômetro com a posição vertical do punho e com o dedo polegar apontado para cima, essa será a posição inicial do jogo onde o tabuleiro sofrerá nenhuma inclinação. A partir dessa posição o jogador deverá realizar as inclinações do punho (esquerda, direita, flexão e extensão além das combinações desses movimentos) para efetuar o controle dos servomotores e explorar os movimentos fisiológicos do punho: supinação, pronação, desvio ulnar e desvio radial

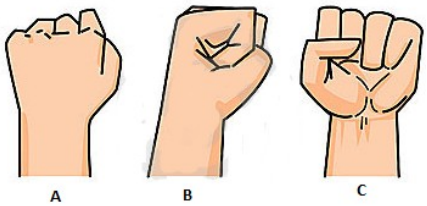

Figura 7: (a) Movimento de rotação do punho pra direita (b) Posição inicial do jogo (c) Movimento de rotação do punho pra esquerda. Observação: o polegar deverá estar estendido.

Haverá 4 servomotores, cada um localizado no meio de cada aresta do tabuleiro. Caso seja realizado, por exemplo, o movimento (a) da figura 7, o servomotor "A", ilustrado na imagem abaixo, atuará inclinando o tabuleiro para a direita. Caso seja efetuado uma combinação de movimentos, como por exemplo, rotação para esquerda e extensão do punho, os servomotores "B" e "D" atuarão inclinando o tabuleiro na diagonal. Essa será a mecânica do jogo.

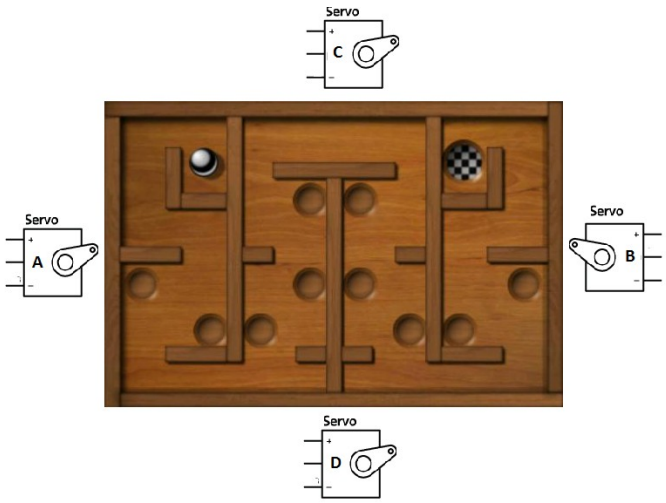

Figura 2: Disposição dos servomotores no tabuleiro.

O tabuleiro será confeccionado em madeira MDF bem como o labirinto e os obstáculos, que serão planejados para que o jogo possa explorar e trabalhar diferentes movimentos de punho com diferentes amplitudes.

V. MATERIAIS UTILIZADOS

- 4 Servomotores TowerPro 9g SG-90;
- Módulo Acelerômetro/Giroscópio MPU-6050;
- MSP430G2553;
- Jumpers macho e fêmea;
- Madeira MDF;

VI. DESCRIÇÃO DE HARDWARE

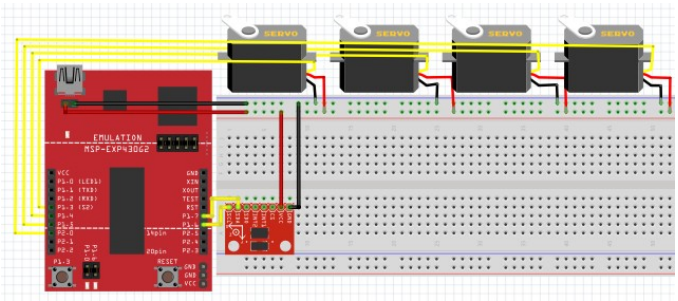

Figura 3: Esquemático de ligação dos componentes eletrônicos com o MSP430.

Para a alimentação do circuito, soldou-se um pino na saída TP1 da placa do MSP430 para obter 5V de alimentação do circuito, proveniente do cabo USB ligado ao computador, visto que os servomotores e o MPU-6050 operam com esse nível de tensão. Conectou-se os pinos SDA e SCL do MPU-6050 nos pinos P1.6 e P1.7 do MSP430 respectivamente, que são os pinos capazes de realizar a comunicação I2C, além disso, conectou-se os fios de controle dos servomotores nos pinos P1.3 ate P2.0 .

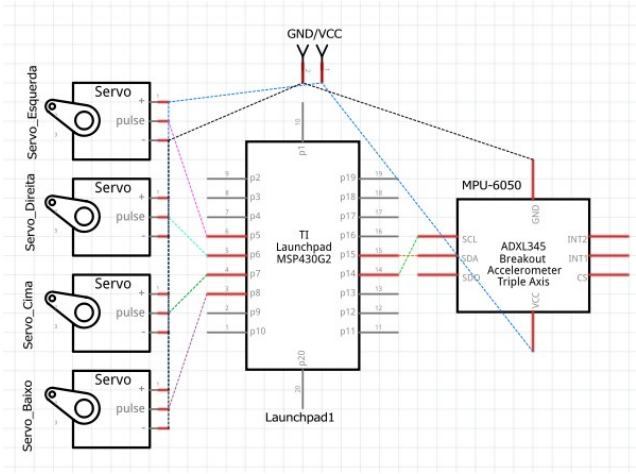

Figura 4: Esquemático das ligações.

Após finalizar os testes dessa parte do projeto, decidiu-se incrementá-lo acrescentando um sistema de feedback para avaliar o avanço dos pacientes, para isso, elaborou-se um circuito para mostrar o tempo que o paciente demora para solucionar o labirinto. O circuito consiste em três displays de sete segmentos, dois para mostrar os segundos e um para os minutos.

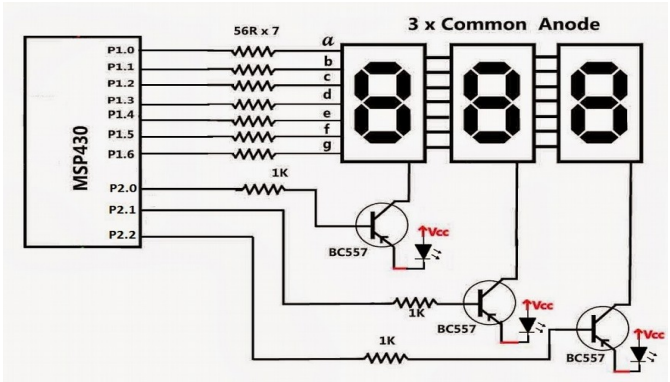

Figura 5: Esquemático de ligações do sistema de feedback.

Além disso, haverá um botão para: iniciar a contagem, pausar e resetar que ficará próximo do controle de mão.

VII. DESCRIÇÃO DE SOFTWARE

Para os primeiros testes do protótipo, utilizou-se o software “Energia” para desenvolver o software embarcado. A linguagem de programação utilizada foi C. A programação foi baseada no seguinte diagrama de blocos do sistema:

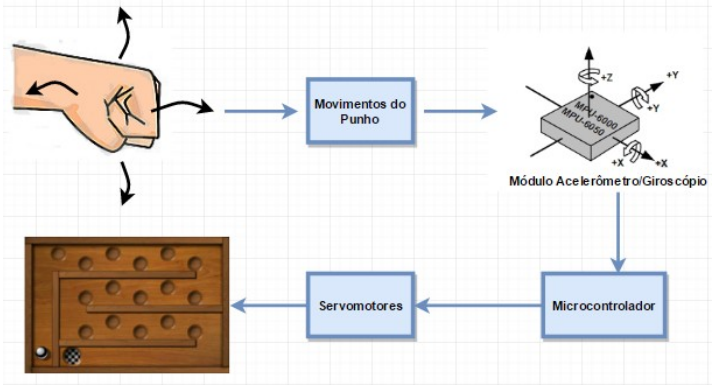

Figura 6: Diagrama de blocos do sistema.

Para estabelecer a comunicação I2C entre o microcontrolador e o módulo MPU-6050 utilizou-se a biblioteca “Wire.h” e para controlar os servomotores a “Servo.h”.

```
#include <Wire.h>
#include <Servo.h>

const int MPU_addr=0x68;  // Endereço I2C do MPU-6050
int16_t AcX,AcY;
Servo Servo_Esquerda;
Servo Servo_Direita;
Servo Servo_Cima;
Servo Servo_Baixo;

int val_esq;
int val_dir;
int val_cima;
int val_baixo;
int preVal_esq;
int preVal_dir;
int preVal_cima;
int preVal_baixo;
```

Figura 7: Descrição de software, primeira parte

Em 1, é feito a inclusão das bibliotecas utilizadas, em 2, é declarado as seguintes variáveis: “MPU_addr” que armazenará o endereço I2C do MPU-6050 (0x68) e “AcX, e AcY” que armazenarão os valores obtidos pelo acelerômetro nos eixos X e Y respectivamente, além disso é definido as entidades dos 4 servomotores. Em 3, é declarado as variáveis que atualizarão constantemente a posição dos servomotores.

```
void setup()
{
  Wire.setModule(0);
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);

  Serial.begin(9600);

  Servo_Esquerda.attach(6);
  Servo_Direita.attach(5);
  Servo_Cima.attach(7);
  Servo_Baixo.attach(8);
  Servo_Esquerda.write(135);
  Servo_Direita.write(135);
  Servo_Cima.write(135);
  Servo_Baixo.write(135);
}
```

Figura 8: Descrição de software, segunda parte.

Nessa parte do código são feitas as configurações iniciais do sistema. Em 4 é definido o protocolo de configuração do módulo MPU-6050, em 5 é definido a taxa de transmissão serial por segundo, em 6 são instanciados os pinos de ligação do MSP430 com os servomotores e em 7 é configurado a posição inicial dos servomotores (135º).

```
void loop(){
  Wire.beginTransmission(MPU addr);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr,14,true);

  AcX=Wire.read()<<8|Wire.read();
  AcY=Wire.read()<<8|Wire.read();
  //AcZ=Wire.read()<<8|Wire.read();
  //Tmp=Wire.read()<<8|Wire.read();
  //GyX=Wire.read()<<8|Wire.read();
  //GyY=Wire.read()<<8|Wire.read();
  //GyZ=Wire.read()<<8|Wire.read();

  val_esq = map(AcX, 0, -17000, 135, 40);
  val_dir = map(AcX, 0, 17000, 135, 40);
  val_cima = map(AcY, 0, 17000, 135, 40);
  val_baixo = map(AcY, 0, -17000, 135, 40);
```

Figura 9: Descrição de software, terceira parte.

A partir dessa parte do código as instruções entram em um “loop” e são constantemente repetidas. Em 8 é realizado a inicialização da comunicação I2C do MSP430 com o MPU-6050, em 9, as variáveis “AcY” e “AcX” recebem do MPU-6050 os valores do acelerômetro no eixo Y e X, em 10, as variáveis (val_esq, val_dir, val_cima e val_baixo) recebem os valores captados pelo acelerômetro do MPU-6050 nos eixos X e Y (AcX e AcY), positivo e negativo (-17000 até 17000) já mapeados no intervalo de angulação dos servomotores (0 a 180º) graças a função “map” cuja sintaxe é: map(valor, mínimo_atual, máximo_atual, novo_mínimo, novo_máximo) em que:

- Valor: Número a ser mapeado, geralmente armazenado em uma variável (AcX e Acy);
- Mínimo_atual: Limite inferior do intervalo de valores atuais (0);
- Máximo_atual: Limite superior do intervalo de valores atuais (-17000 ou 17000);
- Novo_mínimo: Limite inferior do novo intervalo de valores mapeados (135º);
- Novo_máximo: Limite superior do novo intervalo de valores mapeados (40º);

Do jeito que foram estabelecidos os parâmetros da função "map" garantiu-se que cada motor inclinaria de acordo com a direção e sentido de movimento do punho.

```
if (val_esq != preVal_esq)
  {
    if(val_esq>135){
      val_esq = 135;
    }
      Servo_Esquerda.write(val_esq);
      preVal_esq = val_esq;
  }
```

} 11

Figura 10: Descrição de software, quarta parte.

Em 11, é definido uma estrutura condicional para: atualizar constantemente a variável "preVal_esq" de acordo com novos valores armazenados em "val_esq"; enviar o valor atual de "val_esq" para o servomotor (Servo_Esquerda) e impedir que esse servomotor efetue inclinação superior a 135º, para evitar que o tabuleiro perca contato com os servomotores. Essa estrutura condicional se repete ao longo do código, porém, para os outros pares de variáveis e servomotores: val_dir e preVal_dir para o servomotor Servo_Direita, val_cima e preVal_cima para o servomotor Servo_Cima, val_baixo e preVal_baixo para o servomotor Servo_Baixo.

```
Serial.print(" | AcY = "); Serial.println(AcY);
Serial.print("AcX = "); Serial.println(AcX);
```

} 12

Figura 11: Descrição de software, quinta parte.

Por último, em 12 é feito a visualização, no monitor serial do software "Energia", dos valores do acelerômetro mediante a comunicação serial.

Após validar os primeiros testes no software "Energia" usando as bibliotecas disponíveis decidiu-se migrar o código para o software "Code Composer" devido à possibilidade de executar o código instrução por instrução, o que facilita encontrar possíveis erros, além disso foram feitas alterações no código a fim de otimizá-lo, para isso, as bibliotecas e instruções com um nível de abstração elevado foram substituídas por instruções com um nível de abstração menor, no qual a definição de variáveis e o mapeamento das portas I/O são feitos manipulando os registradores do microcontrolador.

O diagrama de blocos abaixo, resume as etapas de funcionamento do código:

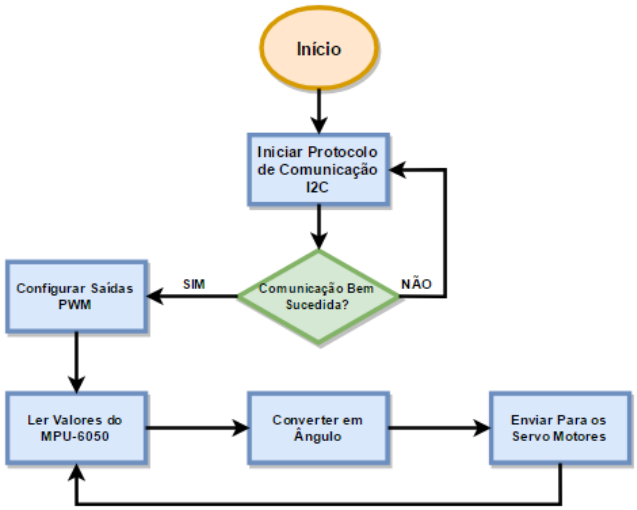

Figura 12: Diagrama de blocos do código.

O código inicia tentando efetuar o protocolo de comunicação I2C com o MPU-6050, esse tipo de protocolo opera no modelo "mestre-escravo" em que um dispositivo atua como mestre (MSP430) e os demais como escravo, neste caso o MPU6050. A função do mestre é coordenar a comunicação, ou seja, é ele quem envia as informações a determinado escravo ou obtém informações.

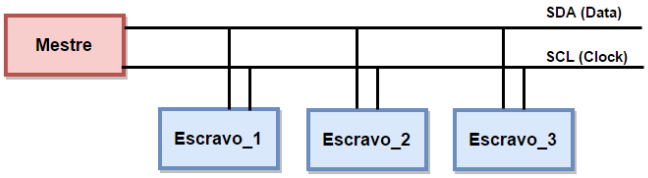

Figura 13: Esquemático de ligações da comunicação I2C

Os dispositivos I2C possuem um endereço, geralmente de 7 bits e expresso em hexadecimal, que os identifica, o MPU6059, por exemplo, possui endereço 0x6. Além desses bits de endereçamento, existe o oitavo bit para identificar quando é uma operação é de escrita ou de leitura (read/write).

Figura 14: Bits de endereçãmento do dispositivo escravo.

Após inicializar o protocolo I2C, é feito a configuração das saídas PWM para controlar os servomotores, para isso, utilizou-se os TIMERs disponíveis no MSP430G2553 (TIMER0 e o TIMER1) O TIMER0 possui dois registradores de captura disponíveis (TA0CCR0 e o TA0CCR1) já o TIMER1 possui três (TA1CCR0, TA1CCR1, TA1CCR2) Os registradores TAxCCR0 possuem um endereço de interrupção de maior prioridade e, portanto, são eles que controlam o funcionamento dos TIMERs de forma geral. Além disso existem dois registradores responsáveis por configurar o modo de funcionamento dos TIMERs: o TA0CTL e o TA1CTL.

Para gerar as saídas PWM configurou-se TIMERs no modo "Up" (MC_1) para que o parâmetro de parada da contagem sejam os valores armazenados em TAxCCR0 e no modo de comparação para que os sinais de saída sejam alterados por hardware quando a contagem chegar no valor armazenado nos registradores de captura (modo 7 de operação) gerando assim, o sinal PWM, conforme a imagem abaixo:

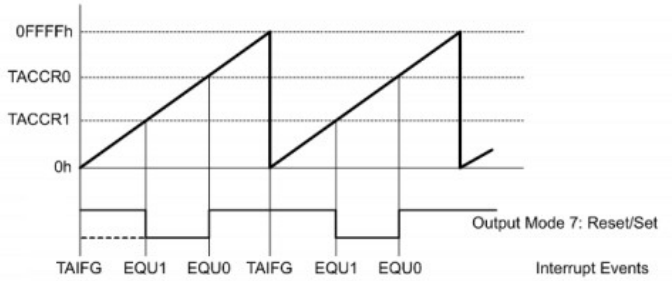

Figura 15: Sinal de saída com base nos valores armazenados nos registradores de captura.

O servomotor utilizado neste projeto é controlado com um sinal com frequência de 50 Hz e pulsos com larguras de 1 até 2 ms, variando a largura do pulso dentro desses limites é possível alterar a posição do servo em até 180º, conforme observa-se na imagem abaixo:

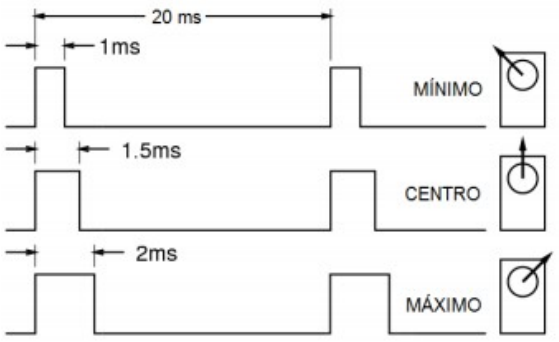

Figura 16: Sinais de controle do servomotor.

Os valores obtidos pelo MPU-6050, são convertidos pela função map, já citada anteriormente, porém, nesta nova versão do código decidiu-se por elaborar a função map, conforme a imagem abaixo, e não utilizá-la pronta como foi feito na versão anterior.

```
long map(long X, long in_min, long in_max, long out_min, long out_max)
{
  return (X - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

Figura 17: Descrição operacional da função map.

Os parâmetros de entrada da função são equivalentes aos citados na versão anterior do código. Após feito a conversão, esses valores são repassados para os registradores dos TIMERs, responsáveis por gerar o sinal PWM com largura de pulso entre 1 e 2 ms.

DESCRIÇÃO DO SOFTWARE DO SISTEMA DE FEEDBACK

O timer0_A foi configurado de forma que aconteça uma interrupção a cada 100ms, e a cada interrupção uma variável inteira "unidade" é incrementada. Como os display acionam com nível baixo, as saídas foram todas predefinidas com nível alto no início do programa. Em um loop infinito está a multiplexação dos displays e o envio do sinal de saída para os segmentos, na forma de switch-case para cada dígito dos displays. O switch-case que varia de acordo com a "unidade" tem 11 casos, para valores de 0 a 10, de forma que cada um dos casos de 0 a 9 envia o digito para os displays, e o caso 10 retorna o valor da "unidade" para 0 e incrementa a variável "dezena". O mesmo ocorre com as variáveis "dezena" e "centena", mas no caso da "dezena" há incremento da "centena", e no caso da "centena" não há incremento de outras variáveis. Em cada switch-case as saídas P2.X são alternadas de forma que os displays apaguem e no fim de cada switch-case o display correspondente é ligado pelas portas P2.X.

VIII. DESCRIÇÃO DA ESTRUTURA

Para confecção da estrutura adquiriu-se uma caixa de madeira MDF com dimensões: 21 cm de comprimento, 21 cm de largura e 8,5 cm de altura onde foram posicionados os 4 servomotores no centro de cada aresta da caixa. Para o tabuleiro, onde ficará o labirinto, também utilizou-se madeira MDF com dimensões: 16 cm de comprimento, 16 cm de largura e 2 cm de altura que foi posicionado sobre os servomotores, conforme as imagens abaixo:

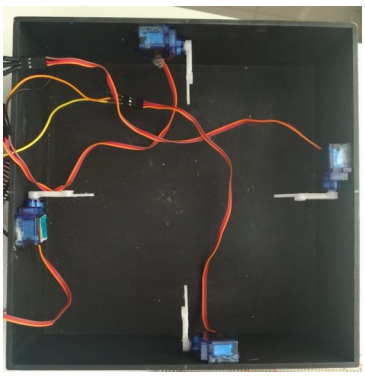

Figura 18: Estrutura do projeto visto de cima.

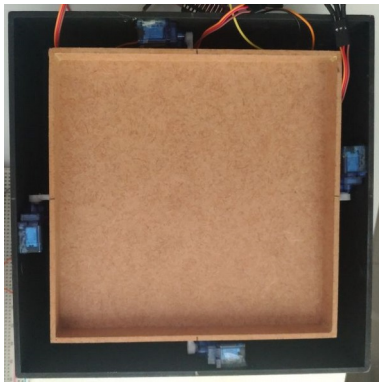

Figura 19: Estrutura do tabuleiro visto de cima.

VIII. CONCLUSÕES E POSSÍVEIS MELHORIAS

- Colocar algum material sobre o tabuleiro para aumentar o atrito e explorar mais os movimentos do punho;
- Estabelecer níveis de dificuldade;

REFERÊNCIAS

[1] Ana Paula Salles da Silva, "Os jogos de movimento e as práticas corporais na percepção de jovens" Tese de pós graduação em Educação Física, 2012.

[2] Erik Nardini Medina, "Realidade virtual pode tornar sessões de fisioterapia mais estimulantes" Reportagem da Revista Inovação, 2016.

[3] BRIO catalog. Retrieved 2011-02-14.

[4] Rodrigo Flausino, "Os jogos eletrônicos e seus impactos na sociedade", 2008.

[5] ACE, Gestão em Saúde, "Manual de Goniômetria Medição dos Ângulos Articulares"

[6] Marques, Amélia Pasqual – Manual de Goniômetria – 2. Ed. Barueri, SP: Manole, 2003. ISBN 85-204-1627-6

[7] Davies, J. H – MSP430 Microcontroller Basics .