UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ (UTFPR)
CURSO DE ENGENHARIA DE COMPUTAÇÃO

MATHEUS V. B. TURATTI
JADER F. HEREDIA
RENALTON A. C. FILHO

SIMULAÇÃO DA FÍSICA DE ARREMESSO COM INTERFACE TFT E MATRIZ DE LEDS

OFICINA DE INTEGRAÇÃO 1 – RELATÓRIO FINAL

CURITIBA

2019

MATHEUS V. B. TURATTI
JADER F. HEREDIA
RENALTON A. C. FILHO

SIMULAÇÃO DA FÍSICA DE ARREMESSO COM INTERFACE TFT E MATRIZ DE LEDS

Relatório Final da disciplina Oficina de Integração 1, do curso de Engenharia de Computação, apresentado aos professores que ministram a mesma na Universidade Tecnologica Federal do Paraná como requisito parcial para obtenção da aprovação na disciplina.

Orientador: Prof. Dr. Gustavo Benvenutti Borba
Prof. Dr. Ronnier Frates Rohrich

CURITIBA

2019

Este trabalho é dedicado ao professor professor Dr. César Manuel Vargas Benítez, cuja pura felicidade de brincar com eletrônicos insipira os alunos nos momentos de falhas...

AGRADECIMENTOS

Agradecemos pela paciência e orientação dos professores Borba e Ronnier que nos ajudaram e guiaram nos momentos de dúvidas, e ao professor Benítez pelas sugestões de tecnologias que só com anos de experiência saberíamos da existência.
Um projeto é mais recompensador quando nasce da vontade de trazer uma solução para uma necessidade, a existência dos professores impulsionam com a experiência nos diversos problemas que surgem ao longo da jornada.

RESUMO

. SIMULAÇÃO DA FÍSICA DE ARREMESSO COM INTERFACE TFT E MATRIZ DE LEDS. 34 f. Oficina de Integração 1 – Relatório Final – Curso de Engenharia de Computação, UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ (UTFPR). Curitiba, 2019.

Os fundamentos teóricos que englobam a eletrônica moderna são diversos, para entendê-los é necessário dedicação, interesse e resiliência, a proposta do projeto de simulação da física de arremesso com interface TFT e matriz de LEDs lança um desafio modesto para os integrantes, relacionando diversas tecnologias de baixo nível para, desta união, obter um jogo coeso e completo.
A proposta de associar tecnologias se resume a utilizar um display com sensibilidade ao toque, para que através de informações espaciais coletadas no toque, seja possível simular uma mecânica de arremesso, no qual LEDs adjacentes se acendam em busca de um alvo aceso na matriz, sendo que o arremesso dependa unicamente da forma que o usuário realiza o movimento sobre o display.
O elemento de jogabilidade é a motivação do arremesso, portanto é necessário considerar como o movimento do usuário é traduzido no trajeto sobre a matriz de LEDs, todo o hardware precisa estar contido numa única caixa, para tanto representar um produto completo, como servir de suporte para que o usuário possa segurar o display.

Palavras-chave: matriz, TFT, arremesso

ABSTRACT

. TFT INTERFACE AND LED MATRIX PITCH PHYSICS SIMULATION. 34 f. Oficina de Integração 1 – Relatório Final – Curso de Engenharia de Computação, UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ (UTFPR). Curitiba, 2019.

The theoretical foundations of modern electronics are diverse, to understand them it is necessary dedication, interest and resilience, TFT interface and LED matrix pitch physics simulation project proposal poses a modest challenge for the team members, linking various low-level technologies to achieve a cohesive and complete game.
The purpose of associating technologies comes down to using a touch-sensitive display, so that through spatial information collected on touch, it is possible to simulate a throwing mechanic in which adjacent LEDs light up in search of a lit target in the matrix. the pitch depends solely on how the user moves the display.
The element of the gameplay is the motivation of the pitch, so it is necessary to consider how the user's movement is translated in the path over the LED matrix. All the hardware must be contained in a single box, to represent a complete product and to offer as support, so that the user can hold the display.

Keywords: matrix, TFT, pitch

LISTA DE FIGURAS

LISTA DE SIGLAS

TFT	Display LCD TFT 2.4”Touchscreen Shield para Arduino
LEDs	light-emitting diode
Arduino	Arduino Uno
I2C	Inter-Integrated Circuit
GND	Ground
CI	Circuito Integrado
PCB	Printed Circuit Board

SUMÁRIO

1 INTRODUÇÃO

Simulaç.ão da **F**ísica de **A**rremesso com interface TFT e matriz de LEDs (SFA), se inspira em tecnologias atuais que já fazem parte do cotidiano, como o simulador de realidade aumentada do jogo Pokemon Go e tecnologias de dois monitores, no qual existe um controle artificial pelo usuário em um dispositivo que consequentemente simula uma resposta em outro ambiente, gerando satisfação para o usuário ao simular um cenário de ação e consequência.
A ideia proposta busca gerar um jogo em que seja possível trabalhar com diversas tecnologias de *hardware* e *software* com fins educativos e que seja possível gerar um produto final que traga satisfação para os envolvidos no projeto.

1.1 MOTIVAÇÃO

A comunicação entre diferentes *devices* que obedecem um único *software* é uma estratégia de difícil arquitetura, pois todos os diferentes componentes de *hardware* teriam de ser profundamente integrados fisicamentes, uma estratégia para contornar esse problema é promover apenas a comunicação entre *devices* e na hora de arquitetar o projeto ele é separado de forma modular, tendo de respeitar sempre o fluxo de informações que estão sendo transmitidas, um paradigma atual no campo de *software*, chamado de programação reativa, existe para contornar esse específico tipo de problema, neste trabalho, porém, não foi necessário utilizá-lo.

1.2 OBJETIVOS

O objetivo ao final do trabalho é ter um jogo simples e completo, no qual o usuário faz um comando de arremesso e a simulação de percurso que as luzes fazem ao acender provoque um sentimento de coerência no usuário e que a troca de informações entre diferentes *devices* seja *seamless* para o usuário.

1.2.1 OBJETIVO GERAL

Utilizando um *display*, desejamos captar algum comando e quando um comando se resume a um movimento de arremesso, ele deve ser reproduzido numa matriz de LEDs para simular, através de luzes se acendendo e apagando, um arremesso feito pelo usuário.
Para motivar os arremessos, alguns LEDs são acesos e eles servem de alvo, o projeto não busca replicar movimentos complexos criados pelo usuário no *display*, ele busca apenas replicar um arremesso de forma consistente com o mundo real.
Uma ideia visual inicial do projeto foi criada conforme a figura 1 e serviu de inspiração para a montagem final.

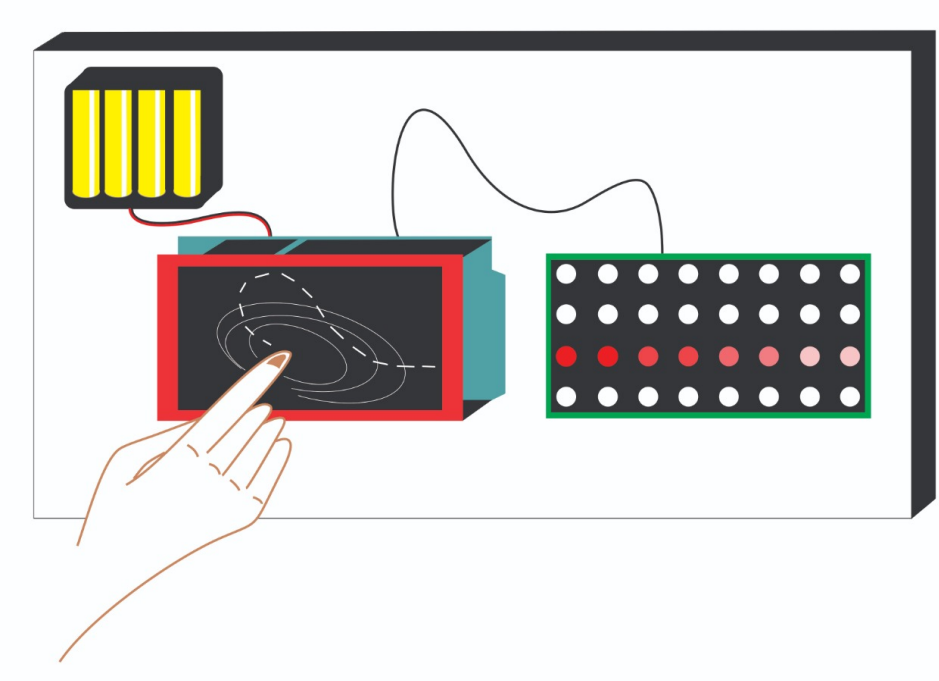


Figura 1: Ideia visual inicial

1.2.2 OBJETIVOS ESPECÍFICOS

O projeto consiste num *display touch screen* próximo a uma matriz de LEDs de 8x4 (32 LEDs total), sendo o dispositivo alimentado por pilhas. O usuário deve arrastar o dedo sobre o *display* em uma direção específica com o objetivo de acertar um ponto identificado previamente na matriz por um LED aceso.
O arremesso, caso passe as checagens que foram coletadas pelo *hardware* em questão de sentido e velocidade, fará com que se acenda alguns dos LEDs em determinada ordem adjacente a fim de simular um arremesso, sendo a luz acesa a simulação do objeto arremessado,

a velocidade de transição dos LEDs dependerá dos dados coletados pelo *display*, a proposta é estabelecer uma relação dependente entre velocidade coletada e velocidade de transição de LEDs.

Para trazer um aspecto verdadeiro de jogo, decidimos criar um sistema de alvos que mudam com uma velocidade, essa velocidade de alteração do alvo depende de um nível que o jogo se encontra, o controle desse nível deve depender dos acertos do usuário e um *display* de 7 segmentos deverá indicar para o usuário em qual nível ele se encontra.

2 FUNDAMENTAÇÃO TEÓRICA

O objetivo deste capítulo é comentar e discutir as referências que foram utilizadas como base para construir o projeto, sendo que ele surgiu inicialmente como um desejo de realizar um jogo específico que atendia condições que os membros levantaram; problemas iam surgindo ao longo do processo de confecção, sendo o papel dos professores essencial para indicar componentes específicos que sanariam nossos problemas.

Apesar de ser um projeto altamente customizado, os *hardwares* utilizados no projeto são ferramentas generalistas muito comuns no campo da eletrônica, portanto, existe uma vasta quantidade de material literário que poderia ser usado de apoio.

Inicialmente não possuíamos recursos para a captação de dados via *touch*, tendo em vista que o Arduino foi um componente que já tinha sido definido para o processamento de informação e distribuição de instruções, optamos pela escolha de um Arduino Shield que proporcionava coleta de dados *touch*, o *display* TFT foi utilizado, sendo que a única parte que cabia ao grupo para o manuseio do mesmo era calibrar as extremidades via *software* do Arduino(ALMEIDA, 2018). Uma vez que optamos pelo *Shield* e o Arduino, seria necessário uma segundo forma de processamento dos dados coletados, pois o Arduino conectado ao TFT não teria uma forma de entregar as informações para o controle de 32 diferentes LEDs.

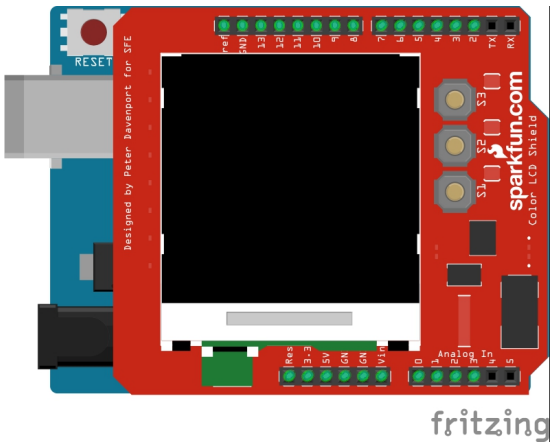


Figura 2: Exemplo do acoplamento TFT Shield

Portanto, foi utilizado uma passagem de informações via protocolo I2C(ARDUINO, 2019a),

uma comunicação nativa do arduino que utiliza a biblioteca Wire(ARDUINO, 2019b).
O funcionamento utiliza duas conexões e GND comum, conforme exemplo da figura 3, a troca de informações por essa via consiste num sistema de mestre e escravo, em que um Arduino recebe a responsabilidade de ser quem envia informações e um ou mais escravos recebem essa informação, essas determinações de quem envia e quem recebe informações é feita unicamente através do *software*, uma vez estipulado quem enviará as mensagens é enviado um fluxo de *bits* que é decodificado pelo segundo Arduino.

Mesmo com um segundo Arduino disponível ainda não era possível fazer um controle

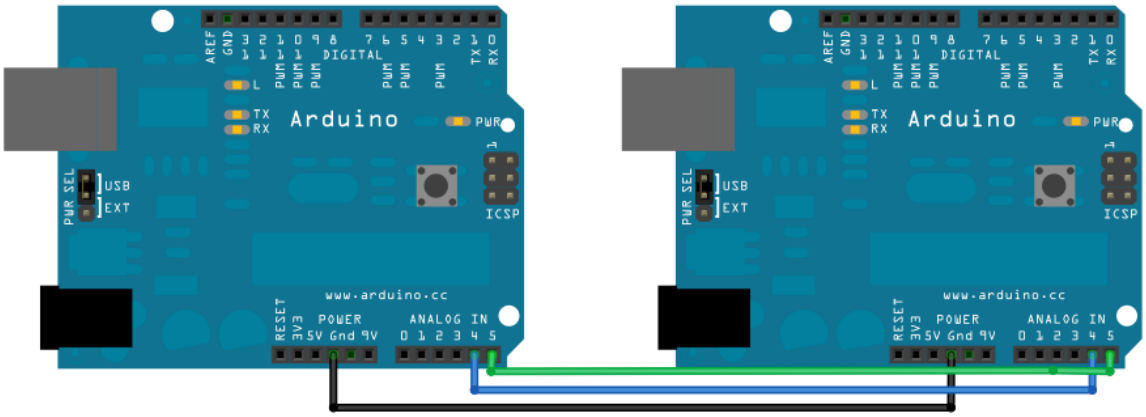


Figura 3: Conexão para I2C, fonte: arduino.cc

individual de LEDs com o Arduino, pois o modelo Uno possui apenas 14 pinos de saída digital, uma saída para esse problema seria adquirir um Arduino Mega, que possui 54 pinos digitais, porém optamos pela utilização de registradores devido ao barateamento do projeto e a oportunidade de trabalhar diretamente com registradores.
Optamos pelos registradores 74HC595N, *serial-in paralel out*, a função dele é receber informações de forma serial, salva na memória *D-Latch* interna e uma vez que a opção *enable* de *output* seja ativada o CI libera de forma paralela 8 informações salvas, além disso ele conta com pinagem para uma montagem sequencial deles, no modelo de pinagem da figura 4, o pino 14 é a entrada que recebe as informações para serem preenchidas na memória *D-Latch*, os pinos de 1 a 7 e o pino 15 são responsáveis pelas saídas da memória.
Preencher um registrador de forma serializada de forma que os *outputs* não fossem controlados traria uma problema para a implementação do projeto, pois a informação precisa atravessar por todos os CIs até chegar num valor desejado, então para chegar no LED de número 30 haveria um rastro de iluminação de percurso e energia seria perdida na iluminação dos LEDs entre 0 e 30, para combater isso, o CI fornece uma opção de *enable*, o diagrama funcional do CI funciona conforme a figura 5.

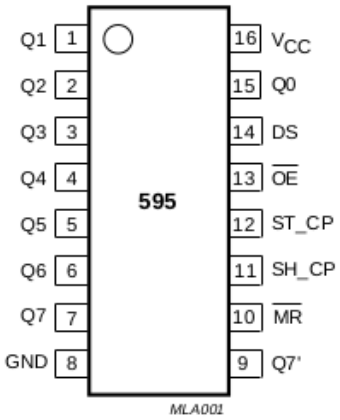


Figura 4: Pinagem 74HC594N, fonte: Philips Semiconductors Datasheet

Os pinos 14, 11 e 10 garantem o preenchimento serial de informações, em que o pino 10 em

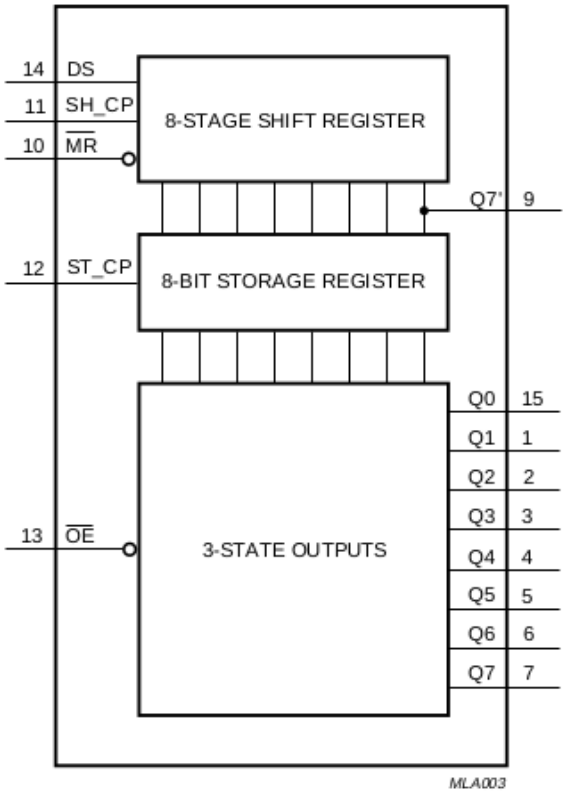


Figura 5: Diagrama funcional CI 74HC595N, fonte: Philips Semiconductors Datasheet

LOW preencherá todos os registros com *LOW*. O pino 14 representa um valor a ser inserido na memória e o pino 11 indo *LOW-to-HIGH* é o gatilho que salva o valor desse pino, uma vez que o pino 12 entrar no modo *LOW-to-HIGH* as informações salvas do pino 14 pelo *clock* do pino 11 serão salvas na memória e o pino 13 em *LOW* é o *enable* que permite a saída paralela de informações.

3 METODOLOGIA

No início do projeto tinhamos apenas a convicação de usar um Arduino e fazer um jogo de arremessos simulados com um display, desde cedo conseguimos dividir as necessidades do grupo em tarefas modulares a serem feitas, definimos então 5 módulos conforme a figura 6.

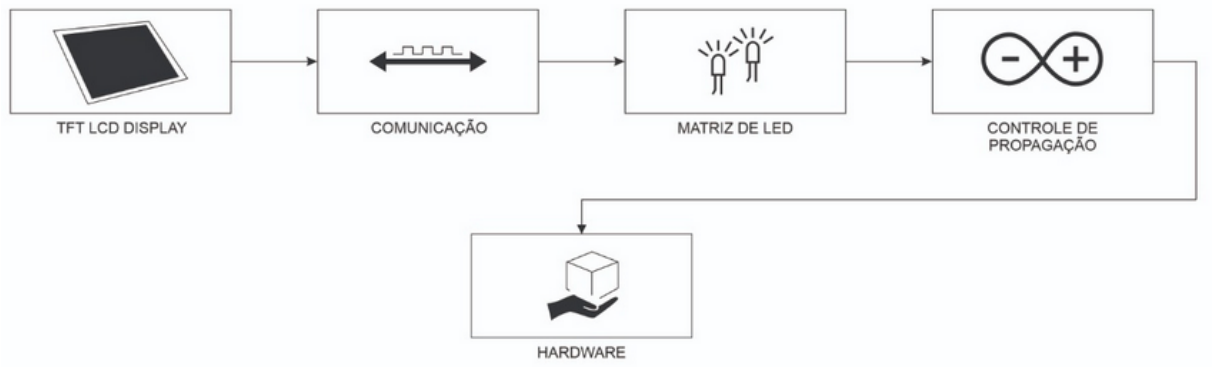


Figura 6: Divisão de tarefas

3.1 VISÃO GERAL

O funcionamento do projeto seguiu o fluxo indicado pela figura 6, começamos o projeto desenvolvendo a recepção dos toques pelo TFT, depois dessa coleta percebemos a necessidade da comunicação pela via I2C, uma vez que conseguimos confirmar que o segundo Arduino recebia as informações corretas foi feita uma montagem simples, figura 7, entre um Arduino e uma matriz de LED através dos CIs 74HC595N, sendo confirmado que era possível controlar com o segundo Arduino passamos para o estágio de integração, no qual utilizando os dados enviados pela coleta feita pelo TFT controlaríamos padrões de movimento sobre a matriz de LEDs, assim que chegamos num padrão que os membros se sentiram satisfeitos, começou a fase da montagem do *hardware*, em que foram soldados os componentes e esses encapsulados dentro de uma *case* que protege o *hardware* e a fonte de alimentação externa.

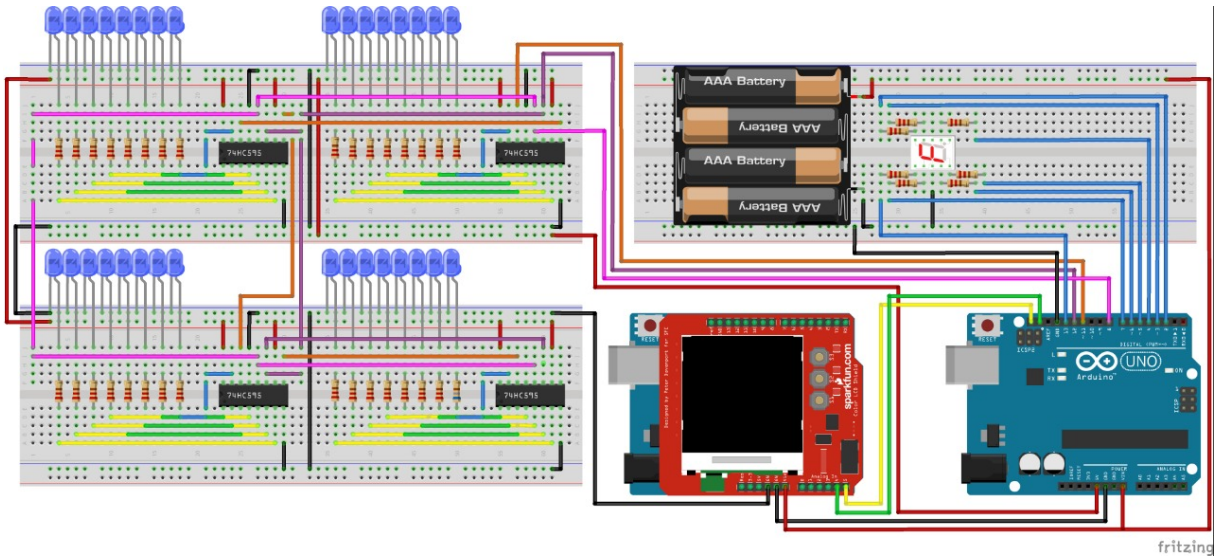


Figura 7: Montagem na Protoboard

3.2 PROJETO MECÂNICO

A *case* era essencial para que houvesse uma forma de segurar o jogo como um produto completo e que uma vez na posse do usuário ele pudesse fazer um arremesso sem ter de se preocupar com a conexão entre componentes, para tanto foi feito uma caixa de madeira com espaços abertos de exposição do *display* e o *display* de 7 segmentos que indica o nível dentro do jogo.

Como o jogo era alimentado por uma fonte externa que estaria contida dentro da *case* era necessário considerar um *switch* que controlasse a alimentação do jogo.

3.3 PROJETO DE HARDWARE

O *hardware* precisava ser montado relativamente cedo para fazer testes de arremesso com o *software* do Arduino, a figura 7 representa as montagens que fizemos em protoboard.

Uma vez que os CIs agem de forma serial era necessário realizar uma montagem ordenada, pois essa montagem definia a numeração LEDs que seriam acessados pelo *software*, uma vez realizada a montagem e os testes, concretizamos a esquemática do projeto, figura 8, e então começamos o processo de soldagem.

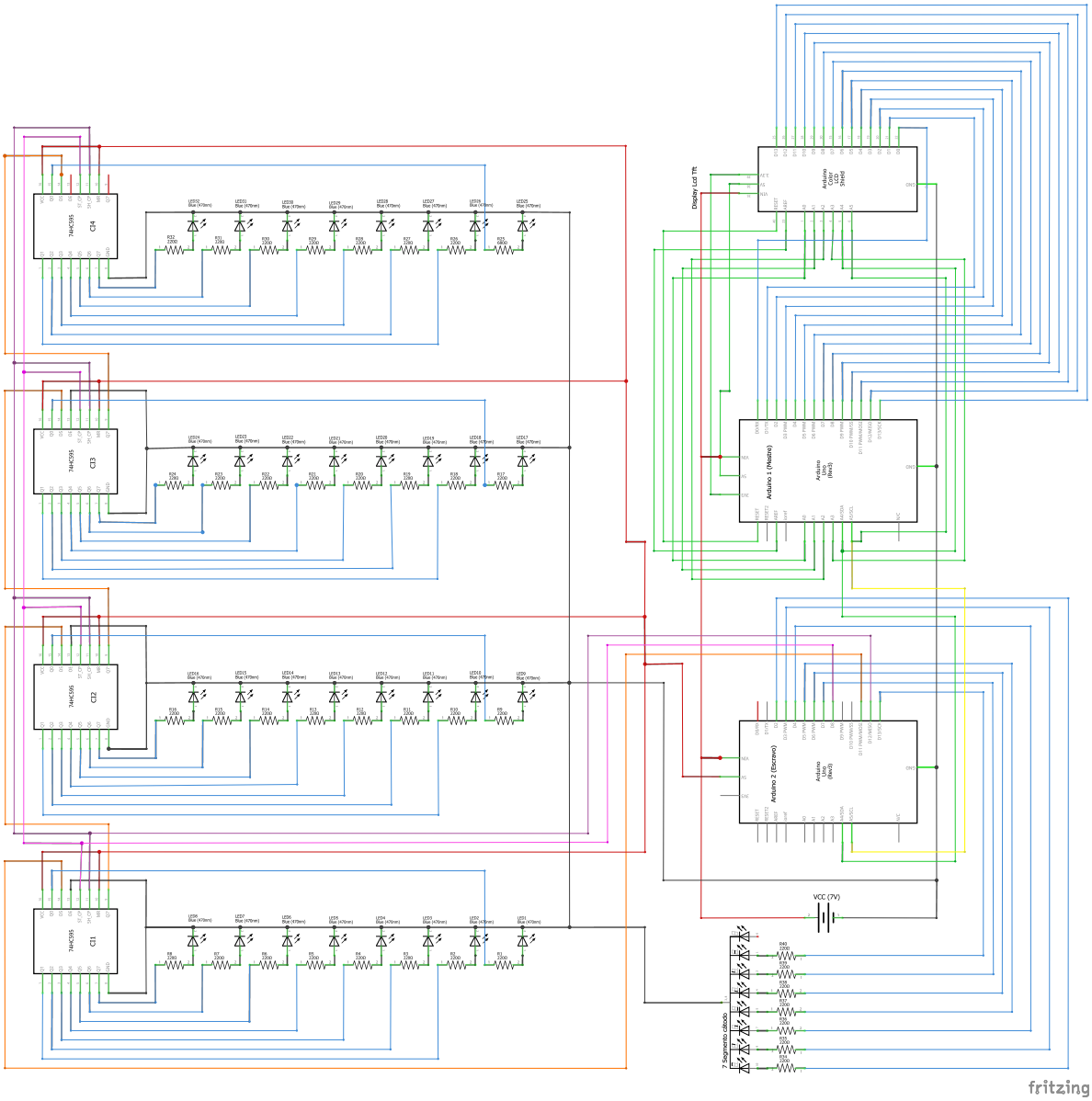


Figura 8: Esquemático do hardware

A disposição matricial dos LEDs levantava um problema de largura, fazer esse projeto com PCB, cada CI precisava de 8 trilhas de *output*, 2 de alimentação e 3 de comunicação, com 4 colunas seriam necessárias 52 trilhas horizontais, tornando inviável a proposta de fazer um projeto completo com um tamanho razoável, portanto optamos pela placa universal, num processo híbrido de trilhas soldadas à mão e soldagens de *jumper*.

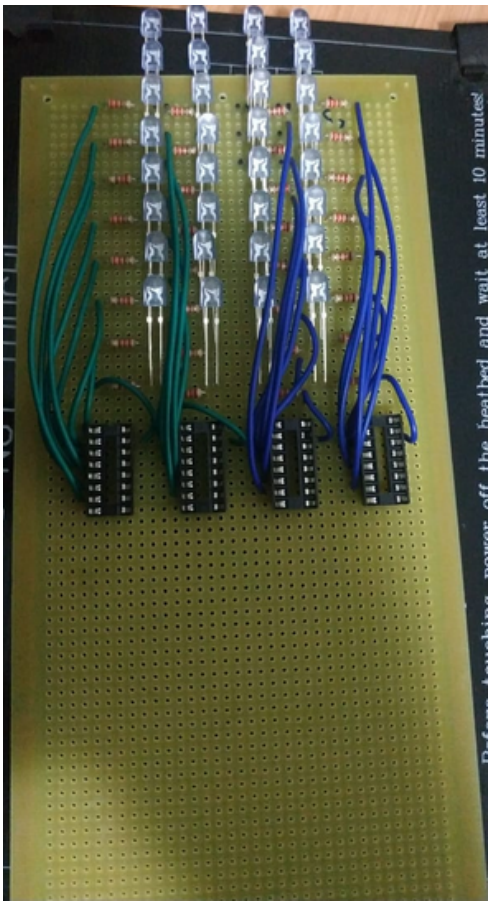


Figura 9: Soldagem, visão dianteira

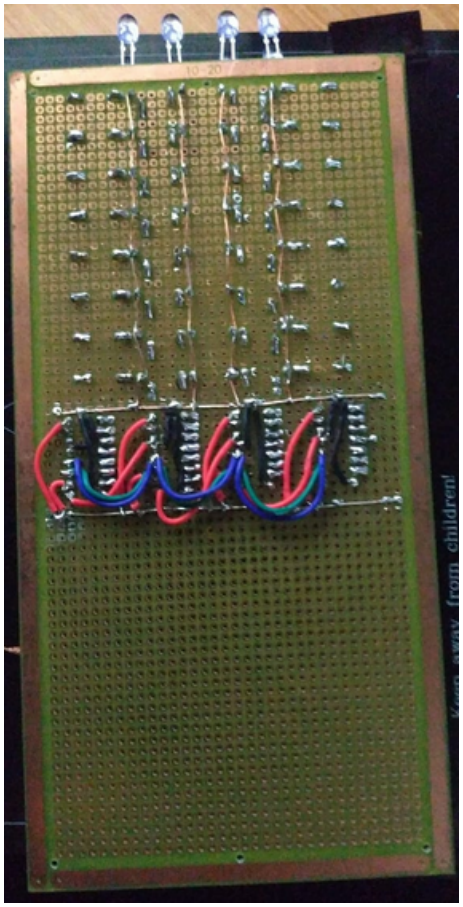


Figura 10: Soldagem, visão traseira

Após os CIs e LEDs soldados, figuras 9 e 10, foram soldados na parte dianteira o Arduino com o TFT e na parte traseira o segundo Arduino.

Além disso fez-se necessário a presença de uma fita isolante líquida, figura 11, o papel dela é evitar algum curto circuito indesejado, a necessidade surgiu no momento em que começou a surgir muitas conexões próximas e que os arduinos repousavam sobre a placa universal.

O *display* de 7 segmentos ficou numa posição deslocada, pois a placa universal já estava sendo ocupada quase completamente, para aproximar as conexões dela com o Arduino de controle, optamos por deixá-la na parte esquerda do TFT.

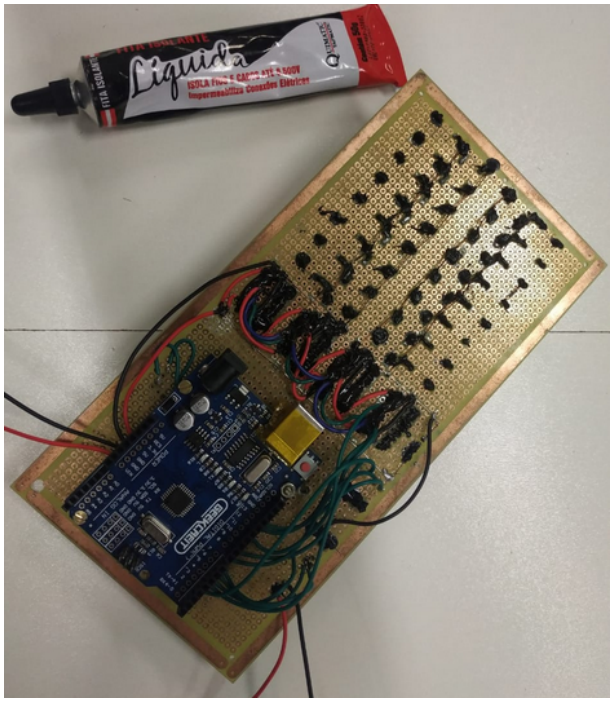


Figura 11: Soldagem, visão traseira final

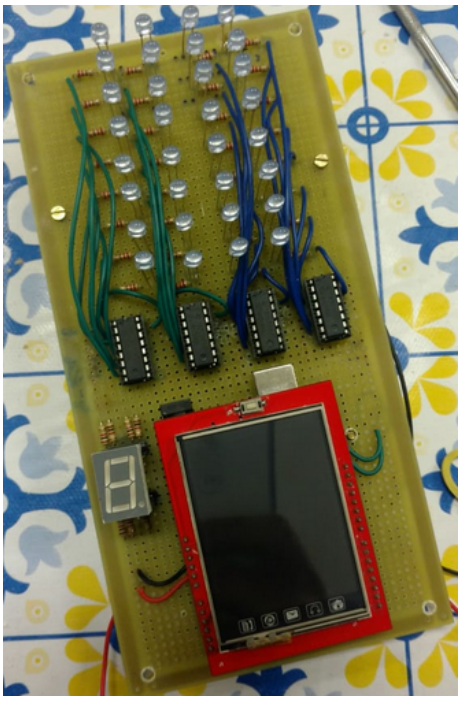


Figura 12: Soldagem, visão frontal final

3.4 PROJETO DE SOFTWARE

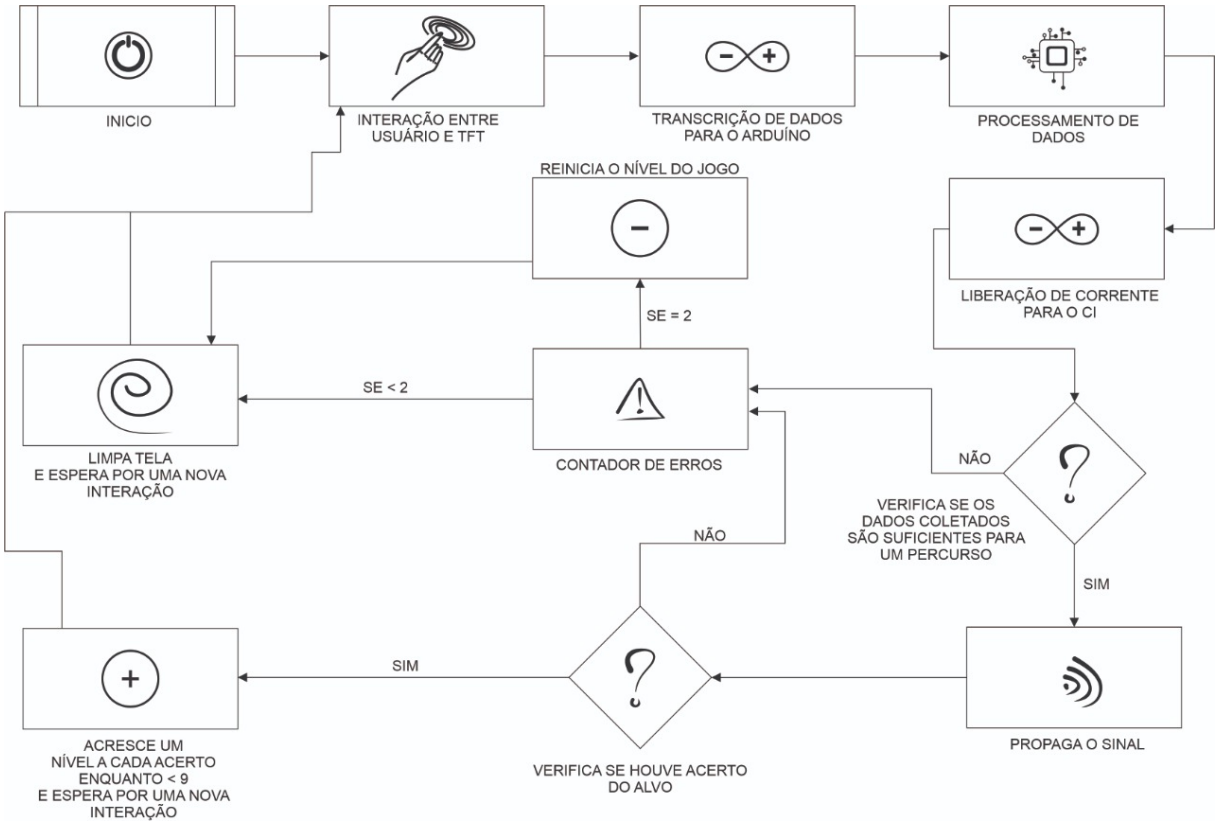


Figura 13: Diagrama de estado

O projeto em *software* é dividido em dois Arduinos, o primeiro deles estava restrito a servir de coleta de dados do toque do usuário, fazer um tratamento desses dados coletados e enviar dados encapsulados que representassem vetores físicos para o segundo Arduino através da conexão I2C entre eles, o segundo Arduino precisava fazer um conjunto de regras que traduzissem os valores coletados para a matriz de LEDs de forma que simulasse um arremesso, além disso ele precisava controlar o *display* de 7 segmentos para representar o nível que o usuário está durante o jogo.

3.4.1 ARDUINO 1

Os dados coletados pelo TFT são representados como três grandezas, x que representa a posição horizontal, y que representa a posição vertical, z que representa a pressão do toque, os valores x e y que representam o mapa cartesiano do *display* do TFT que estão contidos dentro de valores previamente calibrados, figura 14, com base nesses valores de mínimos e máximos foram feitas as regras que determinavam a coluna que o arremesso iniciaria.

As grandezas que queríamos extrair do TFT era o vetor posicional $s = (x,\ y)$ e velocidade $v = (v_x,\ v_y)$, como o Arduino não oferece uma variável de contagem de tempo com início e fim foi necessário utilizar o recurso nativo de $micros()$ que retorna o tempo que o código está em execução, para saber quanto tempo um *snippet* do código leva é necessário chamar o método $micros()$ no início, no final, e tirar a diferença entre os dois, assim tem-se a duração em microssegundos dessa parte do código.

Para iniciar um arremesso foi necessário utilizar um *loop* que checa um valor de pressão z mínimo, porém só realizar um loop dessa checagem gerava erros de valores de lixo de memória do Arduino, portanto foi necessário colocar um $delay()$ forçado dentro do *loop* que servia unicamente para levar mais tempo dentro de cada iteração e como consequência reduzir a quantidade de checagens por instante de tempo, evitando o erro de lixo contido na variável z.

Assim que era detectado uma pressão z válida, o código era deslocado para uma função de arremesso, no qual iniciava-se um contador com $micros()$ para iniciar uma contagem de tempo, a função de arremesso por si só era um *loop*, cada vez que era iterado havia uma nova checagem de pressão, para conferir se ainda havia toque do usuário, caso houvesse uma pressão z válida acima de um valor mínimo, os novos valores x e y eram somados a uma variável e a posição atual era gravada, a partir da segunda iteração iniciava-se o processo de coleta de diferença de posições em x e y, devido a limitações do TFT, uma equação simples de velocidade não era representada bem numericamente devido a inconsistência de coleta de pontos de pressão,

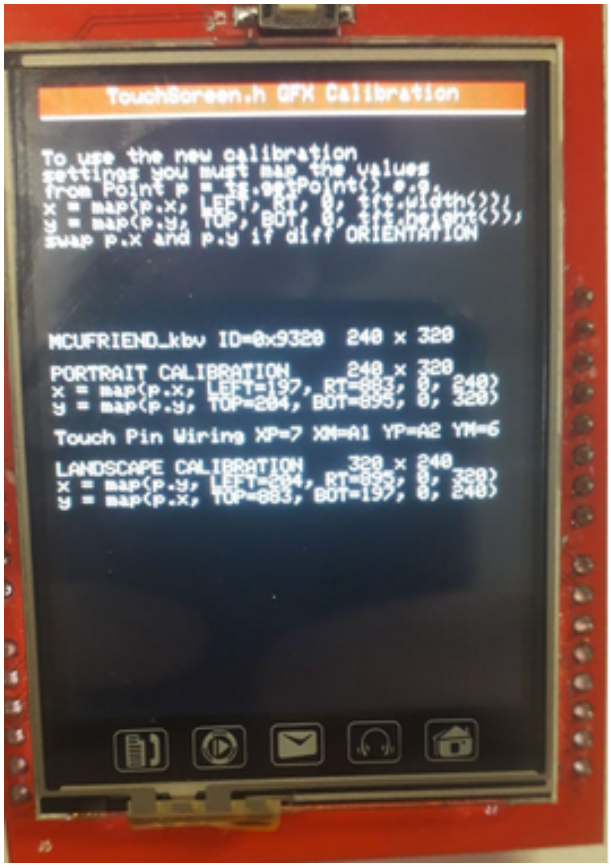


Figura 14: TFT com valores calibrados

portanto a equação que trazia uma representação era uma soma das diversas pequenas mudanças de valor posicional divididos pelo tempo coletado durante a execução do código e esse resultado dividido pela número de iterações feitas pelo loop, sendo que o tempo gasto nos processos de encerramentos eram descontados da variável *tempo*. Com essa equação foi possível captar a diferença entre arremessos, cada iteração dentro da função aumenta o contador e as variáveis totais de v_x e v_y, o sinalizador de saída da função é iniciado quando no início da função a checagem de pressão mínima z não é cumprida, assim inicia-se uma segunda variável de contagem que é incrementada em cada *loop*, caso essa segunda contagem temporal exceda 0.95 segundos, o arremesso é finalizado e os valores são preparados para o envio, esse valor da contagem temporal de saída é descontada da primeira variável de tempo.

Quando o usuário move o dedo para trás optamos por forçar uma reinicialização de todos os valores, então a qualquer mudança negativa do valor de y é como se fosse um novo arremesso.

Uma vez que o arremesso é finalizado a equação de arremesso fica:

$$v_y = \frac{1}{contador * tempo} \sum_{n=1}^{contador} y_f - y_i \tag{1}$$

$$v_x = \frac{1}{2 * contador * tempo} \sum_{n=1}^{contador} x_f - x_i \tag{2}$$

Sendo vetor que representa um híbrido entre média de velocidades e força em y, ao invés de um valor pontual de velocidade, a ”velocidade”de y seria sempre positiva, enquanto a de x pode assumir valores negativos, sendo definido uma velocidade positiva em x um movimento para a direita na matriz e velocidade negativa um movimento sentido a esquerda, um valor de 2 foi inserido após *feedback* visual experimental.
Com os valores calculados era necessário enviá-los pelo canal I2C, porém esse canal só aceitava valores de *byte* como inteiros de 0 a 255, portanto era necessário enviar de uma forma controlada os valores, sendo que a posição de x e y respeitavam a fórmula

$$x = x/255 + x\%255 \tag{3}$$

$$y = y/255 + y\%255 \tag{4}$$

porém o domínio da velocidade de y era sempre inferior a 150 e superior ou igual a 0, sendo enviado sem nenhum tratamento, pois está contido num *byte*. Contudo a variável de velocidade em x podia assumir valores negativos, portanto era necessário criar um terceiro valor que representava uma *flag* entre o valor ser ou não positivo.

$$v_x = (flag)(v_x/255 + v_x\%255) \tag{5}$$

3.4.2 ARDUINO 2

Já no segundo Arduino o fluxo do *loop* fica numa geração de alvos que variam com uma velocidade que depende do nível global que o jogo se encontra, esse nível também define quais saídas do Arduino ficarão em *HIGH* e *LOW* para controlar o *display* de 7 segmentos e fazê-lo representar o nível que está contido em forma decimal.
O fluxo é interrompido assim que o canal de comunicação I2C recebe comunicação, uma vez que isso ocorre o código é redirecionado para a função de tratamento de arremessos, os valores são reconstruídos com base nas equações $3, 4, 5$.
Os valores de x definem a coluna, de forma que as 4 colunas são definidos por um intervalo de valores inteiros de x.
O valor de v_y define quantas linhas o arremesso irá percorrer, o código percorre uma iteração em que cada iteração a velocidade de y sofre uma modificação fixa de atrito que impacta tanto na velocidade de intervalo de tempo entre movimento de LEDs adjacentes, quanto no controle de distância, pois caso v_y não supere um valor mínimo a iteração de movimento acaba e o

```
 *  veja que o valor que eu atribui para a variável defineColumn representa a posição do chão de cada coluna
 *  como o for representa a linha, eu sei que a soma de defineRow e defineColumn define qual posição da matriz o arremesso está
*/
if((x>120) && (x<290)){
  defineColumn = 24;

}
else if((x>=290) && (x<390)){
  defineColumn = 16;
}
else if((x>=390) && (x<580)){
  defineColumn = 8;
}
else if((x>=580)&&(x<720)){
  defineColumn = 0;
}
else{
  //Aqui é um else pra caso x > 720 ou x < 120 (saiu pelas laterais)
  if(mistakeG == 1){
```

Figura 15: Código que controla coluna com base no valor de X

movimento é encerrado. O deslocamento horizontal entre colunas ocorre simultâneamente, este também sofre um atrito no sentido oposto que o reduz, mas para cada iteração do *loop* o valor de v_x é somado a variável x e no início da iteração a mesma checagem da figura 15 é feita, e caso a mudança seja suficientemente grande a posição da coluna será diferente.

A matriz de LEDs é representada por 1 dimensão vetorial de números inteiros de 0 a 31 que controla cada LED individualmente, a contagem começa de baixo pra cima, da direita pra esquerda, ou seja:

31	23	15	7
30	22	14	6
29	21	13	5
28	20	12	4
27	19	11	3
26	18	10	2
25	17	9	1
24	16	8	0

Tabela 1: Valores inteiros que representam a posição dos LEDs na matriz

Os movimentos que serão simulados sobre a matriz de LED dependem de relações matemáticas que são extraídas dessa forma de identificação unidimensional dos LEDs.

As relações matemáticas de movimento se resumem a, norte é $+1$, sul -1, leste é ir uma posição para a esquerda, percebe-se que o diferencial entre os valores da esqueda e direita é uma diferença de ± 8. Sendo o movimento à esquerda é -8, e o movimento à direita $+8$.

A ideia é sempre aplicar um desses valores cardinais e checar:

O valor do movimento seguinte está entre 0 e 31? Caso não, ele saiu da matriz de LEDs.

Uma vez que o arremesso é realizado entra em questão o fator de nível e jogabilidade, no nível 1 até 6 existem dois alvos que mudam aleatóriamente segundo a equação em milisegundos.

$$t = (10 - nível) * 460$$

se encontra. O nível máximo atingível é 9, o jogo avança de nível a cada acerto, fazer um acerto quando já está no nível 9 apenas permanece o nível, um acerto é contabilizado quando a posição sendo iterada pelo arremesso é igualada a posição global que segura a variável posicional de alvos, caso a função de arremesso seja chamada e percorrida sem que essa condicional seja invocada pode acontecer dois cenários, o primeiro deles é quando uma variável de erros ainda está zerada, nesse caso não há mudanças além do fato dessa variável ser alterada de valor para 1, no caso em que a variável de erros já esteja em 1 o nível do jogo é alterado para 1 e a variável de erros é reinicializada em 0, quando um acerto ocorre essa variável de erro também é zerada, ou seja, o nível só volta para 1 quando ocorre dois erros em sequência. Para existir um auxílio visual foram feitas animações na matriz de LED para simbolizar tanto o erro de um arremesso quando o acerto em um dos alvos.

3.5 INTEGRAÇÃO

A integração dos dois foi feita desde o início do projeto, no qual desde a metade do desenvolvimento já existia um protótipo dentro de uma *protoboard*, conforme figura 16.

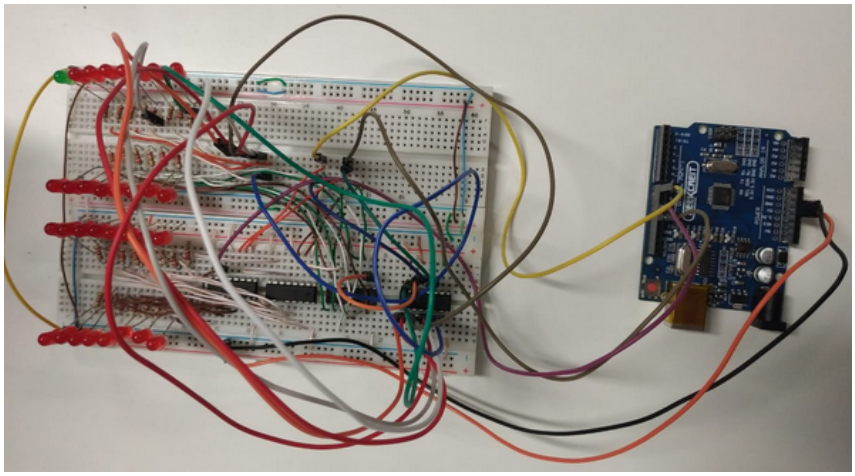


Figura 16: Projeto montado numa protoboard

Sendo essencial para gerar um sistema de *feedback* que foi utilizado para captar a sensação de movimento que mais se adequava a coleta de dados limitadas pela qualidade de *hardware* que tinhamos a disposição.

4 EXPERIMENTOS E RESULTADOS

Após diversos testes percebemos que a coleta de dados do *hardware* era muito limitada, valores em x muitas vezes não faziam sentido e algumas vezes saiam do domínimio do *display*, por isso foi necessário na figura 15 fazer uma divisão de colunas em que o intervalo de valores é desigual entre as colunas.

Durante a criação do projeto utilizamos duas entradas de porta USBUniversal Serial Bus para alimentar os dois Arduinos, contudo uma vez que mudamos para uma alimentação externa de pilhas recarregáveis constatamos que a velocidade de execução do programa é altamente dependente da corrente que a bateria pode fornecer, o que levanta uma questão importante, que funções nativas do Arduino como $delay()$ não são completamente confiáveis, e podemos suspeitar que ao longo do consumo de vida da bateria o desempenho do jogo irá cair e consequentemente prejudicar o fator de jogabilidade.

Após uma série de experimentos concordamos que seria necessário pequenos efeitos de transição e os efeitos que foram escolhidos foram determinados visualmente.

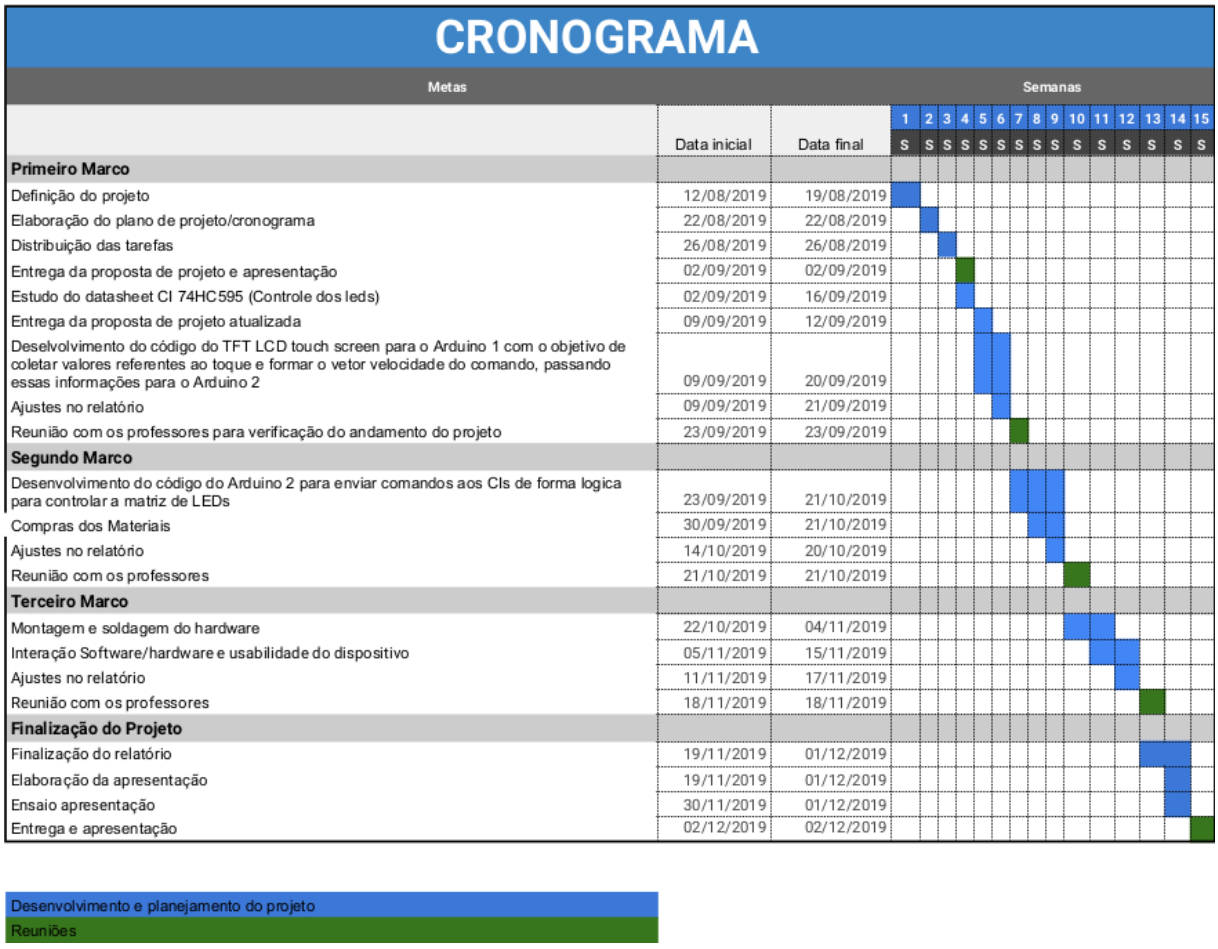
Alguns dos problemas surgiram na montagem em *protoboard* da figura 16, no qual uma das *protoboards* estava com má conexão entre um dos CIs, gerando um erro de difícil diagnóstico que custou extensivo tempo de verificação e remontagem.

Em uma montagem final também houve a queima de um Arduino devido a um curto na hora de fazer a soldagem e o fechamento da *case*.

O vídeo do projeto(FILHO, 2019) com seu correto funcionamento foi criado pelos integrantes como uma forma de acompanhamento visual explicativo do projeto funcionando.

5 CRONOGRAMA E CUSTOS DO PROJETO

5.1 CRONOGRAMA



CRONOGRAMA

Metas	Data inicial	Data final	Semanas 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Primeiro Marco			
Definição do projeto	12/08/2019	19/08/2019	
Elaboração do plano de projeto/cronograma	22/08/2019	22/08/2019	
Distribuição das tarefas	26/08/2019	26/08/2019	
Entrega da proposta de projeto e apresentação	02/09/2019	02/09/2019	
Estudo do datasheet CI 74HC595 (Controle dos leds)	02/09/2019	16/09/2019	
Entrega da proposta de projeto atualizada	09/09/2019	12/09/2019	
Deselvolvimento do código do TFT LCD touch screen para o Arduino 1 com o objetivo de coletar valores referentes ao toque e formar o vetor velocidade do comando, passando essas informações para o Arduino 2	09/09/2019	20/09/2019	
Ajustes no relatório	09/09/2019	21/09/2019	
Reunião com os professores para verificação do andamento do projeto	23/09/2019	23/09/2019	
Segundo Marco			
Desenvolvimento do código do Arduino 2 para enviar comandos aos CIs de forma logica para controlar a matriz de LEDs	23/09/2019	21/10/2019	
Compras dos Materiais	30/09/2019	21/10/2019	
Ajustes no relatório	14/10/2019	20/10/2019	
Reunião com os professores	21/10/2019	21/10/2019	
Terceiro Marco			
Montagem e soldagem do hardware	22/10/2019	04/11/2019	
Interação Software/hardware e usabilidade do dispositivo	05/11/2019	15/11/2019	
Ajustes no relatório	11/11/2019	17/11/2019	
Reunião com os professores	18/11/2019	18/11/2019	
Finalização do Projeto			
Finalização do relatório	19/11/2019	01/12/2019	
Elaboração da apresentação	19/11/2019	01/12/2019	
Ensaio apresentação	30/11/2019	01/12/2019	
Entrega e apresentação	02/12/2019	02/12/2019	

Desenvolvimento e planejamento do projeto
Reuniões

Figura 17: Cronograma original proposto

O cronograma proposto originalmente teve um planejamento bem próximo do alcançável, com uma semana a mais, para cada marco, figura 18, teria sido possível entregar tudo com 100% do estipulado.

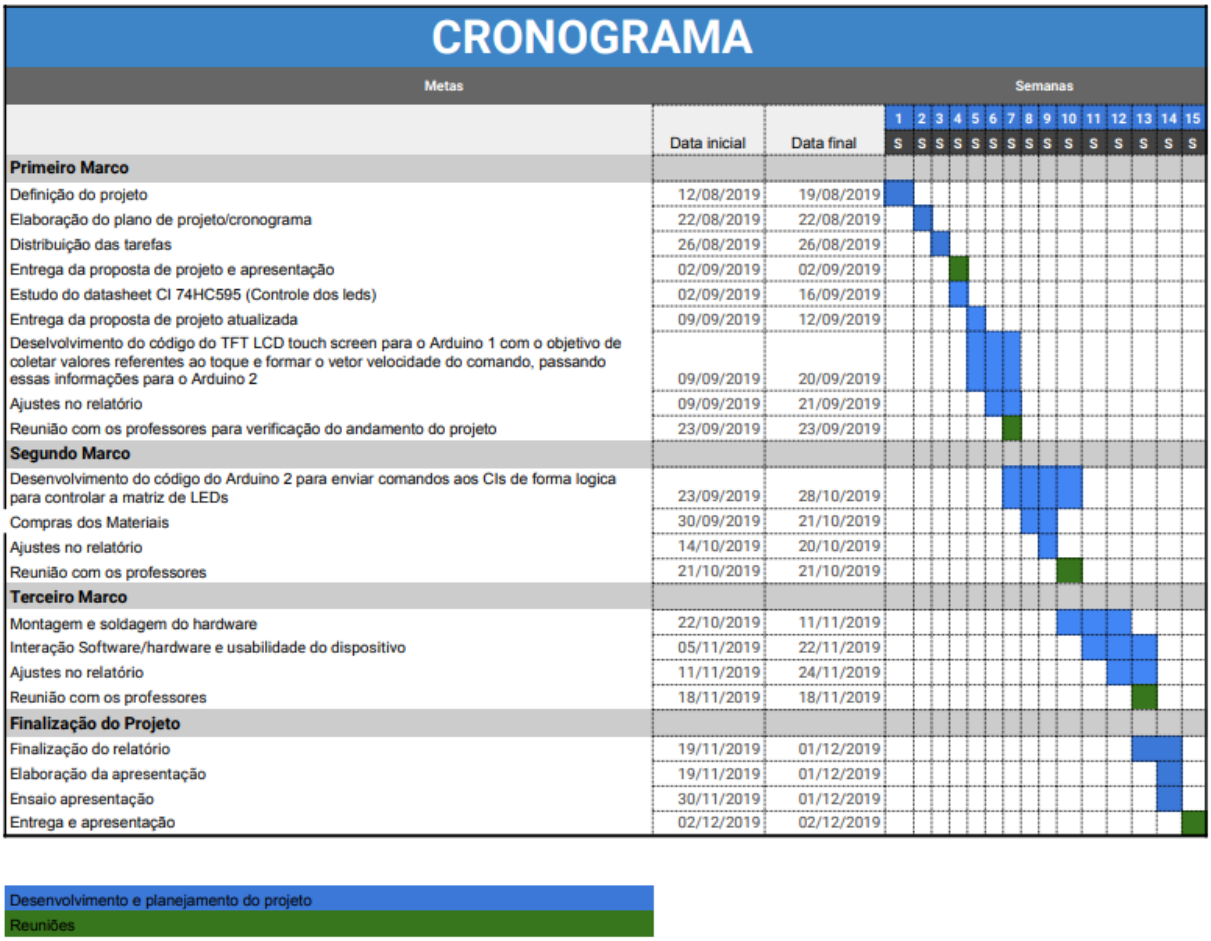


Figura 18: Cronograma original proposto

5.2 CUSTOS

Itens como cabos e *protoboards* os integrantes já possuiam de experimentos anteriores, como fizeram parte de um processo de transição e não foram adquiridos unicamente para essa experiência, não foram listados.

Item	Qtd	Valor(R$)
CI 74HC595	6	10,80
LED difuso vermelho	32	8,00
Resistor 220ohm	60	3,00
LED azul	40	24,00
Arduino Uno	2	48,00
GeekCreit	1	10,00
TFT LCD Shield	1	48,00

Placa perfurada 100x200	1	16,77
Parafuso inox	10	15,90
PO parafuso	5	0,56

Tabela 2: Valores de componentes.

Totalizando R$185,03.

6 CONCLUSÕES

6.1 CONCLUSÕES

A proposta inicial do trabalho foi realizada com êxito, os erros inexperados que foram surgindo ao longo do projeto serviram como uma chance de explorar tecnologias novas paras os integrantes, mas velhas para o mundo da eletrônica, e apesar de existir a opção de utilizar tecnologias já prontas, como uma matriz controlada por periféricos para Arduino, essas são mais caras e tiram a oportunidade de descer em um ambiente de baixo nível para explorar as possibilidades dos tijolos que servem de blocos de criação da eletrônica.
O projeto abriu uma perspectiva da quantidade de trabalho árduo que existe por ”baixo dos panos”e como tarefas simples de hardware podem trazer complicações inesperadas.

6.2 TRABALHOS FUTUROS

O futuro guarda muitos eventos inesperados, a equipe aguarda ansiosamente para trabalhar com embarcados de um nível maior em qualidade e confiabilidade, para que funções mais complexas possam ser computadas.
A metodologia, divisão modular e constantes revisões, adotada pela equipe, foi bastante eficaz e garantiu a entrega final do trabalho sem grandes complicações ou preocupações, acreditamos que essa estratégia de mente aberta e checagem cíclica é necessária para garantir que um projeto se mantenha sempre em andamento e que possa ser finalizado com um produto de qualidade.

REFERÊNCIAS

ALMEIDA, D. **Display LCD TfT 2.4**

Primeiros passos com Touchscreen. 2018. Disponível em: <https://portal.vidadesilicio.com.br/display-lcd-tft-2-4-primeiros-passos-com-touchscreen/>.

ARDUINO. **Master/Slave Arduino**. 2019. Disponível em: <https://www.arduino.cc/en/Tutorial/MasterReader>.

ARDUINO. **Wire library Arduino's Libraries**. 2019. Disponível em: <https://www.arduino.cc/en/reference/wire>.

FILHO, R. A. C. **Simulador de Arremesso com Alvo Projeto para disciplina de Oficina de Integração 1 do curso de Engenharia de Computação.** 2019. Disponível em: <https://youtu.be/FqPi5tf4G_4>.