

RELATÓRIO FINAL

PROJETO DE SISTEMAS EMBARCADOS

PROF. CARLOS HENRIQUE BARRIQUELLO

Gregory de Moraes Rossato¹

Lucas Machado Rister²

Resumo: *Este relatório apresentará um sistema embarcado para conversão de binário para decimal, criado através da ferramenta Arduino Software (IDE), utilizando o microcontrolador Arduino Uno R3 em comunicação com o Display LCD Keypad Shield 16x2. Para o desenvolvimento do sistema, foi implementado a ideia de RTOS (Real Time Operating System), um ambiente multitarefas para sistemas embarcados, através da solução aberta FreeRTOS. A aplicação desenvolvida trata-se de um conversor de binário para decimal, onde a entrada, e saída de dados é realizada via Display LCD Keypad Shield 16x2 e processada pela placa Arduino Uno R3. Então será desenvolvida uma breve introdução a respeito da placa utilizada, o display e o software, além de uma explicação sobre o projeto.*

Palavras-chave: *SISTEMAS EMBARCADOS, ARDUINO, FREERTOS, LCD DISPLAY.*

¹ Acadêmico do curso de Engenharia de Computação da Universidade Federal De Santa Maria- UFSM, matrícula: 201521414, email: gregory.rossato@gmail.com

² Acadêmico do curso de Engenharia de Computação da Universidade Federal De Santa Maria- UFSM, matrícula: 201521115, email: lucas_rm1@live.com

1. INTRODUÇÃO

O projeto tem o objetivo de receber dados de entrada em binário, converter para decimal e mostrar na saída do *display LCD* através de tasks, utilizando o *FreeRTOS*. A intenção do projeto é realizar a conversão de binário para decimal através de um código otimizado.

O código foi implementado, compilado e executado no software *Arduino Software (IDE)* e é, basicamente, a comunicação do *Display LCD Keypad Shield 16x2* com a placa *Arduino Uno R3*.

O projeto apresenta 3 botões de entrada de dados (quando o usuário pressiona o botão direito acrescenta 0 ao valor de entrada, botão esquerdo acrescenta 1 ao valor de entrada e botão cima realiza a conversão de binário para decimal e mostra na tela o valor em decimal). O projeto também possui 3 tasks (input, process e show) que fazem comunicação entre si e 2 filas queue (comunicação entre input e process) e queue2 (comunicação entre process e show).

2. REFERENCIAL TEÓRICO

Neste capítulo serão descritos os componentes usados no projeto e uma breve introdução sobre cada um deles. São eles: *Arduino Software (IDE)*, *FreeRTOS*, placa *Arduino Uno R3* e *Display LCD Keypad Shield 16x2*.

2.1. Arduino Software (IDE)

É um ambiente de desenvolvimento integrado. É um espaço onde você tem tudo, ou quase tudo, que precisa para programar sua placa baseada nesta plataforma, escrevendo seus códigos de forma rápida e eficiente. É uma das opções mais usadas por desenvolvedores de projetos. É um software fácil de se manusear e gratuito. Link do site do software: <https://www.arduino.cc/en/software>

2.2. FreeRTOS

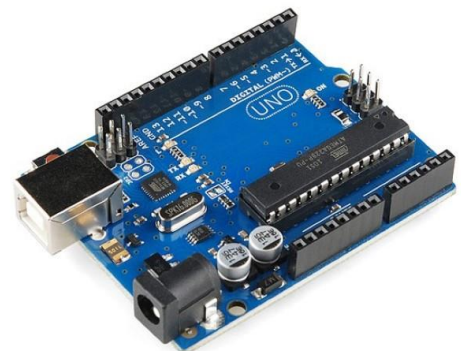
Desenvolvido em parceria com as principais empresas de chips do mundo ao longo de um período de 15 anos, e agora baixado a cada 175 segundos, o *FreeRTOS* é um sistema operacional em tempo real (RTOS) líder de mercado para microcontroladores e microprocessadores pequenos. Distribuído livremente sob a licença de código aberto do MIT, o *FreeRTOS* inclui um *kernel* e um conjunto crescente de bibliotecas adequadas para uso em todos os setores. *FreeRTOS* é construído com ênfase na confiabilidade e facilidade de uso. Disponível em: <https://github.com/FreeRTOS/FreeRTOS-Kernel>

2.3. Arduino Uno R3

Arduino é uma plataforma *open source* ou *hardware* para prototipagem eletrônica, projetada com um microcontrolador *Atmel AVR* com suporte para entrada/saída dados já embutido, com linguagem de programação padrão baseado no em C/C++. O Arduino é utilizado por causa da velocidade que se consegue desenvolver algo, e se é rápido também é mais simples, isso é conseguido devido a modularidade do Arduino, da farta documentação e da grande quantidade de módulos disponíveis para se conectar ao Arduino.

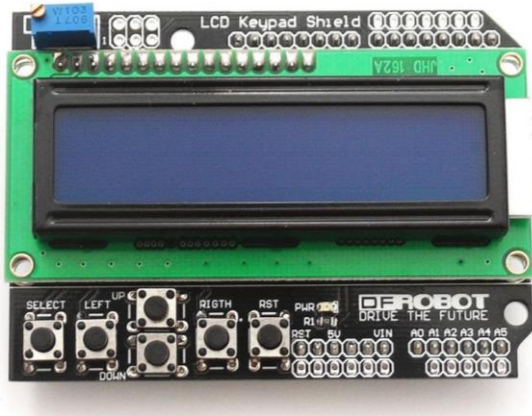
ESPECIFICAÇÃO ARDUINO UNO R3

- **Essa placa não é um Arduino original**
- Microcontrolador: ATmega328
- Tensão de operação: 5V
- Tensão de Alimentação recomendada: 7-12V
- Tensão de Alimentação Limite: 6-20V
- Pinos Digitais I/O: 14 (dos quais 6 podem ser usados como saída PWM)
- Pinos entrada Analógica: 6
- Corrente CC por pino I/O: 40mA
- Corrente para pino de 3,3V: 50mA
- Memória Flash: 32KB(ATmega32u4) dos quais 0,5KB são usados pelo bootloader
- SRAM: 2KB (ATmega328)
- EEPROM: 1KB (ATmega328)



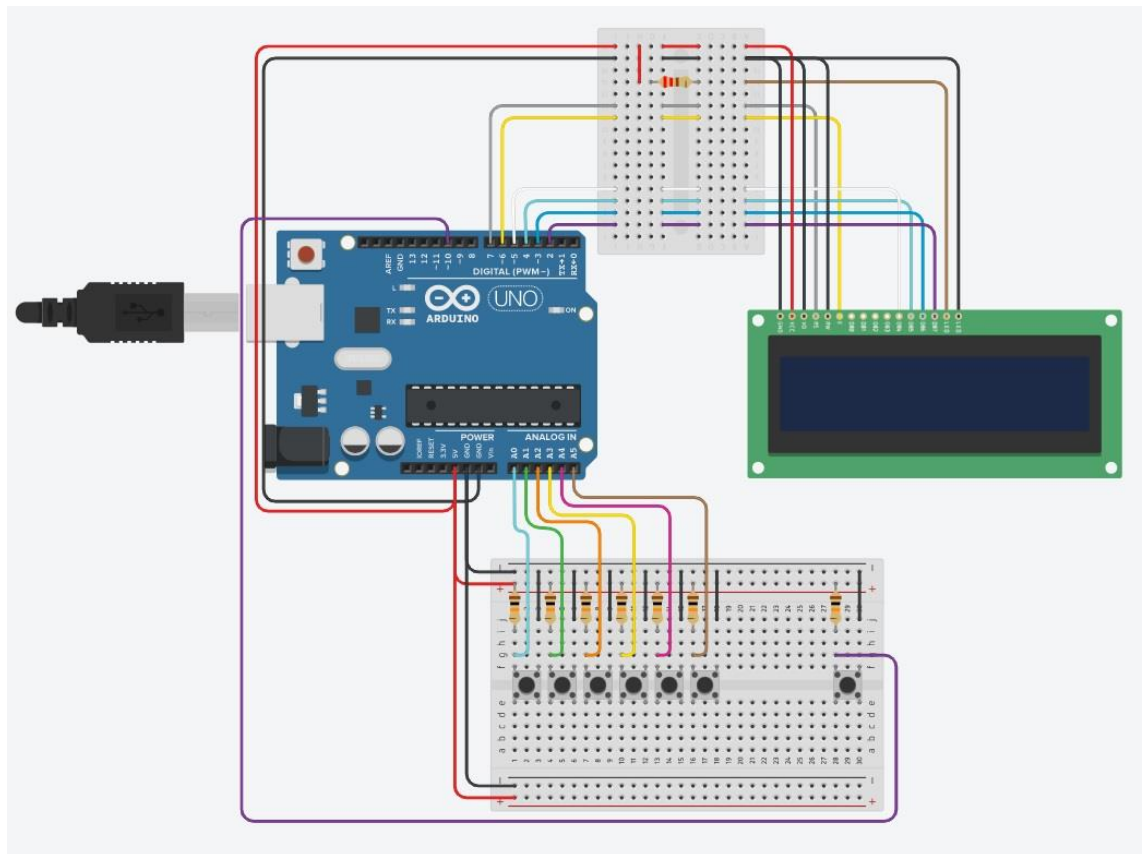
2.4. Display LCD Keypad Shield 16x2

LCD Keypad Shield é um módulo integrado que emprega toda a praticidade dos *Shields*, de um *display* 16x2 e de um conjunto de pequenos botões *push buttons* que são utilizados para navegar em menus na diretamente na tela, com funções selecionar e resetar. Devido ao *LCD Keypad Shield* com botões ser conectado diretamente sobre a face superior do Arduino a possibilidade de falhas nas ligações é nula, além de evitar interrupções de imagens devido a pequenas vibrações no microcontrolador ou na base onde ele encontra-se localizado.



3. DESENVOLVIMENTO

Para desenvolvimento do projeto, inicialmente usamos o *tinkercad*, disponível em: <https://www.tinkercad.com/dashboard>, para familiarização, testes de componentes e código. Após muitas pesquisas e entendimento do funcionamento dos componentes, montamos nosso primeiro esboço de projeto. Segue abaixo a imagem do circuito no *tinkercad*.

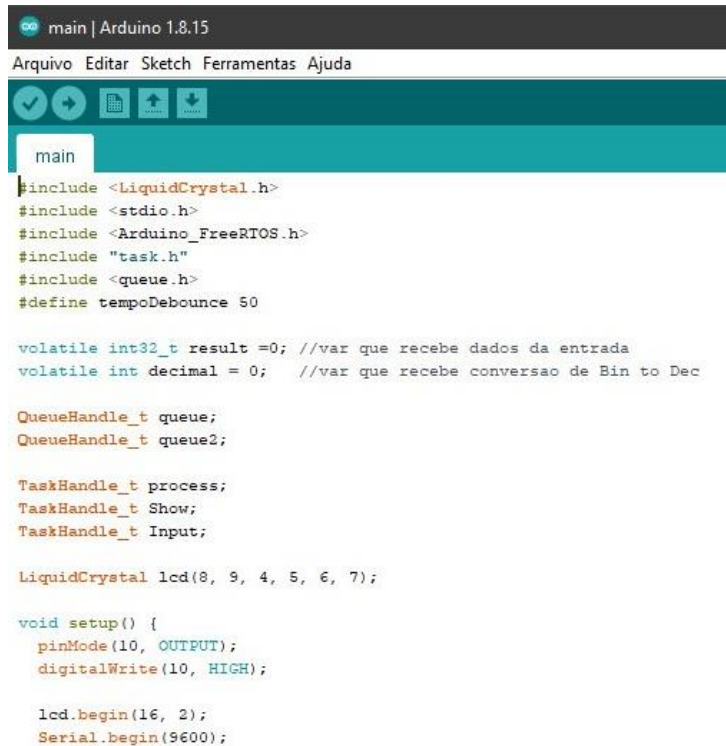


Após nos familiarizarmos e estudarmos também o funcionamento do *FreeRTOS*, começamos a desenvolver nosso circuito e código final.

Para o circuito final, fizemos uma pequena modificação no projeto, ao invés de usarmos o *LCD 16x2*, muitos resistores e botões, escolhemos usar uma *LCD Keypad Shield*. O nosso circuito ficou muito menos poluído, pois o encaixe é perfeito com o *Arduino Uno R3* que usamos, e o nosso código também ficou mais otimizado. Segue as imagens do nosso circuito abaixo. A primeira os componentes individualmente e a segunda com os componentes conectados.



Nosso código foi montado, compilado e executado no *Arduino Software (IDE)*. A seguir serão mostrados trechos do código e uma breve explicação sobre o que está acontecendo. O código encontra-se no repositório *github*, disponível em: <https://github.com/LucasRister/ProjetoFinalEmbarcados>



```
main | Arduino 1.8.15
Arquivo Editar Sketch Ferramentas Ajuda

main

#include <LiquidCrystal.h>
#include <stdio.h>
#include <Arduino_FreeRTOS.h>
#include "task.h"
#include <queue.h>
#define tempoDebounce 50

volatile int32_t result = 0; //var que recebe dados da entrada
volatile int decimal = 0; //var que recebe conversao de Bin to Dec

QueueHandle_t queue;
QueueHandle_t queue2;

TaskHandle_t process;
TaskHandle_t Show;
TaskHandle_t Input;

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

void setup() {
    pinMode(10, OUTPUT);
    digitalWrite(10, HIGH);

    lcd.begin(16, 2);
    Serial.begin(9600);
```

Foram utilizadas as bibliotecas tradicionais, destaque para as bibliotecas *LiquidCrystal.h*, que serve para o funcionamento do *LCD* e para a biblioteca *Arduino_FreeRTOS.h*, que serve para o funcionamento do sistema operacional que foi escolhido, no caso o *FreeRTOS*.

O código possui 2 variáveis globais, *volatile int_32_t* (var que recebe dados da entrada) e *volatile int decimal = 0* (var que recebe a conversão de binário para decimal). São definidas como globais porque só são modificadas na função *process*, o *display* só mostra elas.

QueueHandle_t queue e *QueueHandle_t queue2* são os inicializadores de fila. Detalhe para *queue* que faz a comunicação entre *Input* e *process* e *queue2* que faz a comunicação entre *process* e *Show*.

TaskHandle_t process, *TaskHandle_t Show* e *TaskHandle_t Input* são os inicializadores de tarefas.

LiquidCrystal lcd(8, 9, 4, 5, 6, 7) define a pinagem usada no *LCD*.

pinMode e *digitalWrite* são habilitadores para escrita no *LCD*.

```

static void Input1(void *pvParameters){ //Task para coletar os btns de entrada

int state;
for(;;){
    int valBotoes = analogRead(A0);

    if ((valBotoes < 800) && (valBotoes >= 600)) {
        state = 1; // condicao para o btn SELECT
        xQueueSend(queue, &state, portMAX_DELAY);

    } else if ((valBotoes < 600) && (valBotoes >= 400)) {
        state = 2; // condicao para o btn LEFT;
        xQueueSend(queue, &state, portMAX_DELAY);

    } else if ((valBotoes < 400) && (valBotoes >= 200)) {
        state = 4; // condicao para o btn DOWN;
        xQueueSend(queue, &state, portMAX_DELAY);

    } else if ((valBotoes < 200) && (valBotoes >= 60)) {
        state = 3; // condicao para o btn UP;
        xQueueSend(queue, &state, portMAX_DELAY);

    } else if (valBotoes < 60) {
        state = 5; // condicao para o btn RIGHT;
        xQueueSend(queue, &state, portMAX_DELAY);

    } else {
        state = 0; // condicao para o btn EMPTY;
        xQueueSend(queue, &state, portMAX_DELAY);
    }
}
}

```

Nossa primeira *task Input*. Nela é identificado qual botão foi pressionado e após identificação é enviado para a *task process* através da fila *queue*. As condições para identificar qual botão foi pressionado seguem de acordo com o *datasheet* do *Shield* usado.


```

static void process1(void *pvParameters) // Task para executar as funcoes referente a cada btn
{
    unsigned long delayBotao; //Delay do debounce
    int estadoBotaoAnt = 0; //Var temp para o debounce
    int32_t remainder =0; //var para o resto da divisao do conversor de bin para dec
    int32_t base =1; // var para a base multiplicativa do conversor de bin para dec
    int statel; //var que recebe qual btn foi apertado
    int input =0; //var que determina o que mostrar no display 0 - entrada 1 - conversao

    for(;;){
        xQueueReceive(queue, &statel, portMAX_DELAY);
        if ((millis() - delayBotao) > tempoDebounce) {
            if ((statel != 0) && (estadoBotaoAnt == 0) ) {
                if(statel == 2){ //se btn LEFT pressionado
                    result = result*10 + 1; //recebe 1 e pula para o proximo digito
                    input = 0; // Seta o display para mostrar os dados de entrada
                    xQueueSend(queue2, &input, portMAX_DELAY);
                }
                if(statel == 5){ // se btn RIGHT pressionado
                    result = result*10; // recebe 0 e pula para o proximo digito
                    input = 0; //Seta o display para mostrar os dados de entrada
                    xQueueSend(queue2, &input, portMAX_DELAY);
                }
                if(statel == 3){ //se btn UP pressionado convert Bin to Dec
                    remainder =0;
                    decimal = 0;
                    base =1;
                    while(result > 0){
                        remainder =result % 10;
                        decimal = decimal + remainder* base;
                        result = result/10;
                        base = base*2;
                    }
                    input = 1; // seta o display para mostrar a conversao de bin para dec
                    xQueueSend(queue2, &input, portMAX_DELAY);
                }
                delayBotao = millis();
            }
            if ((statel == 0) && (estadoBotaoAnt != 0) ) {
                delayBotao = millis();
            }
        }
        estadoBotaoAnt = statel;
        input =0; // Seta o display para mostrar os dados de entrada
        xQueueSend(queue2, &input, portMAX_DELAY);
    }
}

```

Acima temos a *task process*, ela recebe pela função *xQueueRecieve* qual botão foi pressionado.

Caso o botão pressionado for o *left*, é acrescentado 1 no valor de entrada e a *flag Input* é setada para 0 (*Input = 0*) e mandada para a *task show* através da fila *queue2*.

Caso o botão pressionado for o *right*, é acrescentado 0 no valor de entrada e a *flag Input* é setada para 0 (*Input = 0*) e mandada para a *task show* através da fila *queue2*.

Caso o botão *up* for pressionado, é realizada a conversão de binário para decimal e a *flag Input* é setada para 1 (*Input = 1*) e mandada para a *task show* através da fila *queue2*.

Se nenhum botão for pressionado, a *flag Input* é setada para 0 (*Input = 0*) e mandada para a *task show* através da fila *queue2*, onde são mostrados os valores de entrada.


```

static void Show1(void *pvParameters){ // Task para mostrar os dados na tela

    int input1 =0;

    for(;;){
        xQueueReceive(queue2, &input1, portMAX_DELAY);
        if(input1 ==0 ){ //Condicao para mostrar os dados da entrada
            lcd.setCursor(0,0);
            lcd.print("Input:");
            lcd.setCursor(2,1);
            lcd.print(result); //Mostra os dados da entrada
        }else if(input1 ==1){ //Condicao para mostrar os dados convertidos
            lcd.clear();
            lcd.print("Dec:");
            lcd.setCursor(2,1);
            lcd.print(decimal); //Mostra a conversao
            vTaskDelay(pdMS_TO_TICKS(5000)); //Deixa os dados na tela por 5s
            result =0; //Reseta a var dos dados convertidos
            lcd.clear(); //Reseta o display
        }
    }
}

```

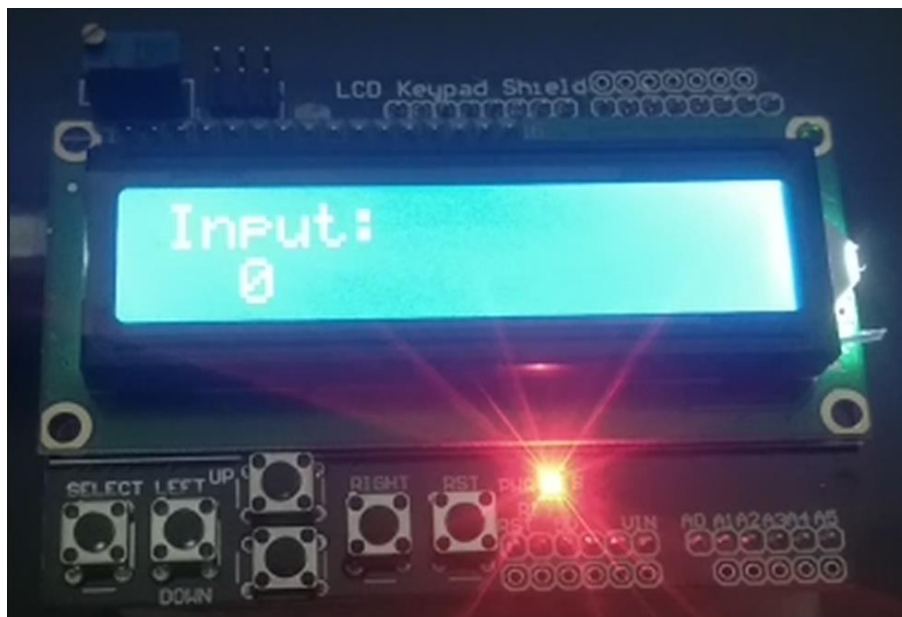
Por último, temos a *task Show*. Ela recebe, através da fila *queue2*, a *flag Input*.

Caso *Input = 0*, os dados de entrada são mostrados no *display*.

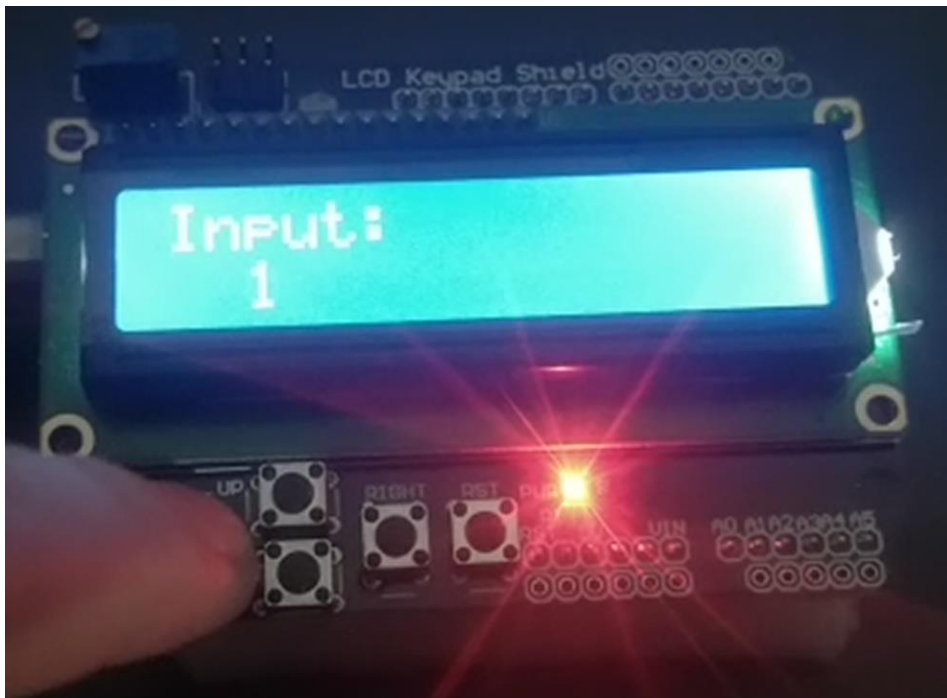
Caso *Input = 1*, a conversão de binário para decimal é mostrada no *display*.

Após a conversão, os dados na tela são mostrados por 5s, após esse tempo os valores de entrada são resetados.

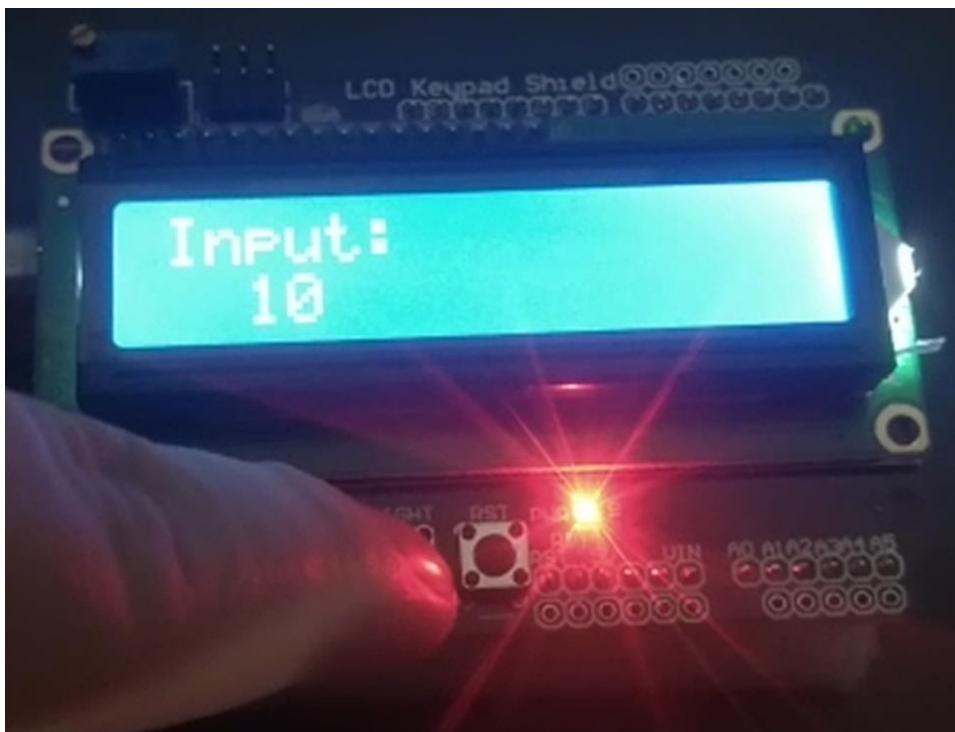
Abaixo segue uma sequência de imagens do nosso circuito físico em funcionamento.



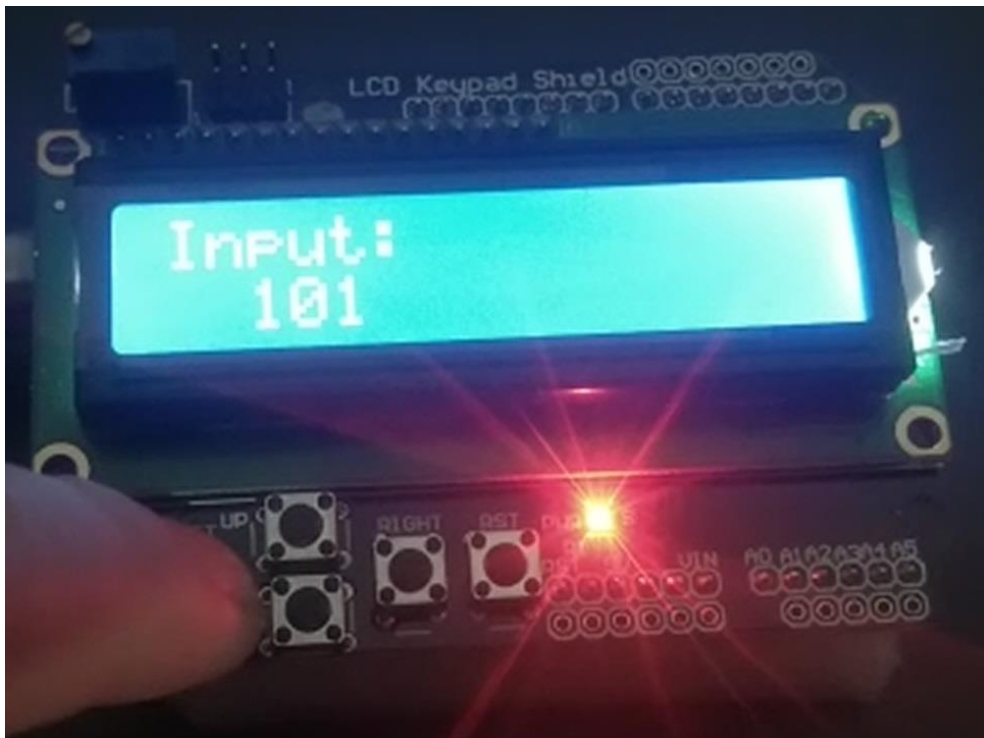
Nenhum botão sendo pressionado, nenhum valor de entrada.



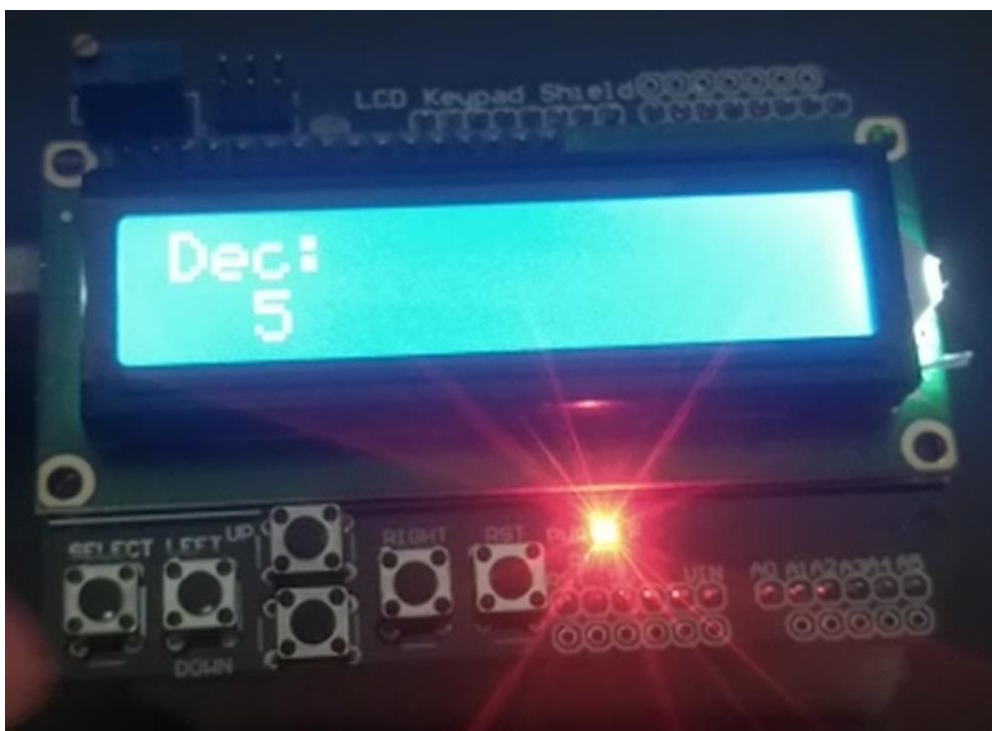
Botão *left* sendo pressionado, é acrescentado 1 ao valor de entrada.



Botão *right* sendo pressionado, é acrescentado 0 ao valor de entrada.



Botão *left* novamente sendo pressionado, é acrescentado 1 ao valor de entrada.



Botão *up* pressionado, a conversão é feita e mostrada no *display* em decimal.



Após a conversão, passou-se 5s e os valores de entrada são resetados.

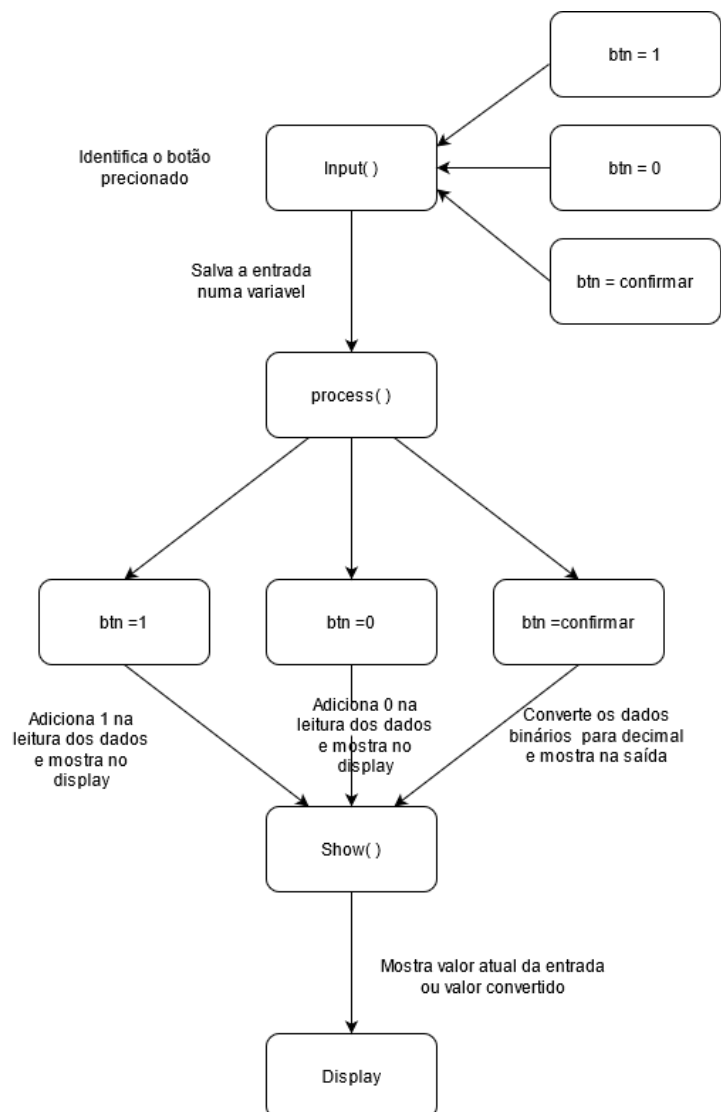


Imagem do Fluxograma do circuito.

4. CONCLUSÃO

O projeto pode servir como base para estudos futuros e mais aprofundados sobre sistemas embarcados, além de permitir o entrosamento do aluno com os materiais envolvidos no projeto.

Através dos estudos realizados, e as informações contidas neste documento, foi possível realizar a elaboração de um projeto de sistemas embarcados para conversão de número binário para decimal usando o sistema operacional *FreeRTOS*.

Para referências futuras este projeto poderá ser aperfeiçoado tornando-o ainda mais útil para a utilização do usuário, trazendo mais opções de conversão de números ou alguma outra operação matemática.

5. REFERÊNCIAS

https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/DFR0009_Web.pdf

<https://linhkientot.vn/ebook/arduino-uno-r3-datasheet.pdf>

<https://www.dcc.ufrj.br/~marcel/wp-content/uploads/2016/10/Introdução-à-Programação-Embarcada-com-Arduino.pdf>

DENARDIN, G. W. e BARRIQUELLO, C. H. Sistemas operacionais de tempo real e sua aplicação em sistemas embarcados.

Repositório do *github*: <https://github.com/LucasRister/ProjetoFinalEmbarcados>