Universidade do Porto

SBMI

Controle de enchimento de tanque com microcontrolador

Cássio Santos up201802025

December 28, 2018

Contents

1 Introdução

Desde do início dos anos 70 que os microcontroladores começaram a estar na atualidade do desenvolvimentos tecnológico tendo ele sido o mesmo um dos grandes avanços tecnológicos do século.

Desde de então, os microcontroladores tem vindo a ser largamente implementada uma vez que, por exemplo, ajudam a substituição de sistemas analogicos antigos e complexos, tornando também a automação de máquinas elétricas mais simples e largamente acessível.

Neste projeto será apresentado através do controle do enchimento de um tanque algumas das potencialidades de um microcontrolador Atmega328P, para o mesmo além do microcontrolador foram utilizados uma gama de sensores e atuadores para demonstrar a versatilidade destes sistemas digitais em adquirir sinais tanto digitais como analógicos para supervisionar e controlar o sistema. Utilizou-se 2 diferentes sensores para a aquisição do nível do tanque tendo eles sido o potenciômetro que varia a sua resistência consoante o nível de uma boia flutuante no topo e parafusos ligados de lado do tanque que caso submersos pela a água assumem o mesmo potencial do pino de referência situado na parte de baixo do tanque.

Entre os atuadores, foram utilizados um motor de bombeamento de água para encher o tanque, dois displays de sete segmentos para supervisionar o nível do tanque, uma fila de oito leds com o mesmo propósito e por fim, um buzzer para sinalizar situações de adversas do tanque (demasiado cheio ou demasiado vazio).

2 Fundamentação Teórica

A seguir, serão apresentados alguns fundamentos teóricos necessários para compreender a metodologia e os resultados obtidos nesse projeto.

2.1 Conversor ADC

Dados analógicos são aqueles que variam de um modo contínuo e gradual. No mundo real, um sensor sente um parâmetro físico e converte em um sinal analógico elétrico equivalente. O processamento de um sinal analógico é bastante ineficiênte em termos de acurácia, velocidade e saída desejada, por isso faz-se necessário converter esses tipos de sinais para digital usando um *Analog to Digital Converter (ADC)*.

Supondo o uso de uma referência de 5V, qualquer valor analógico entre 0 e 5V é convertido para um valor ADC equivalente. O alcance de 0 a 5V é dividido por 2^{10} = 1024 passos. Dessa forma, uma entrada de 0V entregará um valor ADC de saída de 0, uma entrada de 5V será um valor ADC de 1023, 2.5V retornará algo entorno de 512 e assim por diante [1].

3 Metodologia

3.1 Boia e conversor ADC

Para identificar as condições do tanque são utilizados dois tipos de informação, uma proveniente da boia posicionada no topo do tanque e outra proveniente de quatro sensores de nível posicionados ao longo da parede do tanque.

Um sinal analógico é adiquirido ao analisar a variação de resistência obtida pela movimentação do potenciômetro localizado na base da aste da boia. Na Figura 1 é possível ver a disposição da boia em relação ao tanque na qual a posição 1 representa a situação de tanque cheio e a posição 2 representa a posição de tanque vazio.

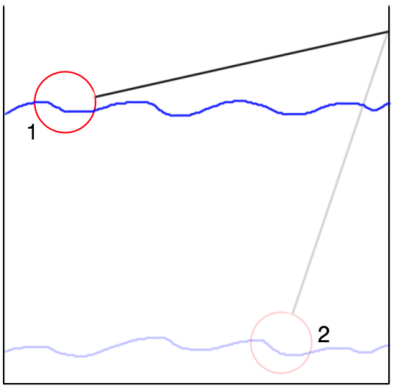


Figure 1: Esquema da boia no tanque

Após o condicionamento do sinal de tensão utilizando um circuito com amplificador operacional *LM747N* (Figura 2), um nível de tensão entre 0 e 5V é lido por um dos pinos com suporte a ADC do microcontrolador. Este sinal é então convertido para um valor analógico entre 0 e 1023 utilizando o conversor ADC embutido do microcontrolador ATmega328p.

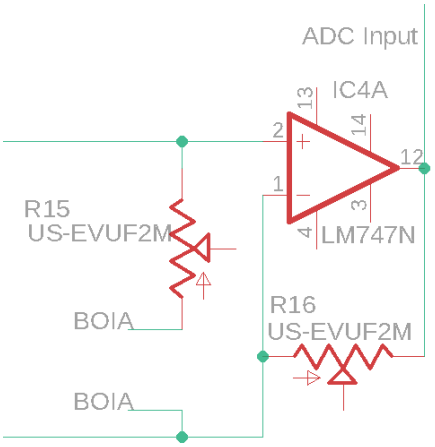


Figure 2: Circuito amplificador de tensão para o sinal ADC

O processo via software para efetuar a conversão ADC consiste na inicialização do ADC (Figura 3) na qual é configurado o modo de operação do conversor, o prescaler e a tensão de referência.

```
/*
 * ADC conversor initialization
 */
void adc_init(void)
{
  /* AVCC with external capacitor at AREF pin */
  ADMUX = (1 << REFS0);
  /* ADEN: 1 => ADC enable; ADPS: 111 => Prescaler 128 */
  ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
}
```

Figure 3: Inicialização do modulo ADC

Para efetuar a leitura do sinal é chamada então a função *adc_read* (Figura 4) que recebe como parâmetro o canal (Pino do microcontrolador), que o circuito multiplexador interno do micro deve selecionar para efetuar a conversão, e então retorna o valor processado.

```c
/*
 * Read Adc value at specified channel
 * - ch: Channel to be read
 */
uint16_t adc_read(uint8_t ch)
{
  ch &= 0b00000111;           // AND operation with 7 to keep value between 0-7
  ADMUX = (ADMUX & 0xF8) | ch; // Clears the bottom 3 bits before ORing

  // Start ADC convertion writing '1' to ADSC
  ADCSRA |= (1 << ADSC);

  // Wait conversion to complete.
  // Conversion complete when ADSC is back to 0
  while (ADCSRA & (1 << ADSC))
    ;

  return (ADC);
}
```

Figure 4: Leitura e conversão do sinal analógico para digital

A este valor são definidos limiares que representam o máximo e o mínimo de água que pode estar presente no tanque. Estes limiares são utilizados para fazer uma correta conversão do nível do tanque (valor adc lido) para uma porcentagem que represente o nível de água atual no tanque em relação ao máximo de água que o tanque pode comportar. A função que define essa conversão pode ser vista na figura 5.

```c
/**
 * Convert a adc value to a percentage based on the tank's thresholds
 */
uint32_t get_floater_percent(void)
{
  uint32_t adc_floater = adc_read(FLOATER);
  if (adc_floater < LOWER_LEVEL)
    return 0;
  else if (adc_floater > HIGHER_LEVEL)
    return 99;
  else
    return (adc_floater - LOWER_LEVEL) * 99 / (HIGHER_LEVEL - LOWER_LEVEL);
}
```

Figure 5: Conversão de valor ADC para nível do tanque em porcentagem

3.2 Sensores de nível

Outra forma de identificação das condições do tanque é utilizando a informação obtida dos quatro sensores de nível posicionados ao longo da parede do tanque (Figura 6). Esses sensores são nada menos que simples parafusos que obtem o mesmo potencial do pino S0 quando estão submersos em água. Na figura 6 por exemplo, os pinos S1, S2 e S3 apresentão um valor de tensão equivalente aquela sendo aplicada no pino S0 pois a água funciona como um condutor nesse caso, equanto isso, o pino S4 tem um valor de tensão flutuante por não estar conectado nem ao *ground (GND)* nem ao pino de referência (S0).

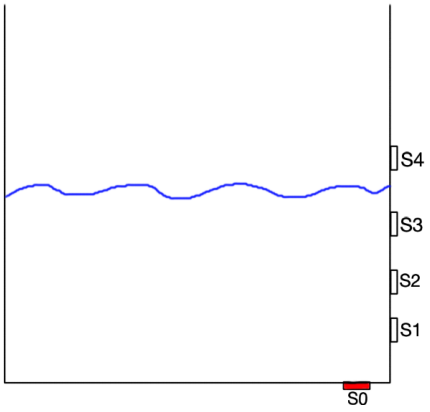


Figure 6: Esquema dos sensores de nível do tanque

Para tratar essas situações de tensão flutuante nos pinos e inverter a lógica de funcionamento das entradas para uma lógica negativa é utilizado um circuito com um inversor *74HC14N* apresentado na figura 7 na qual S1-S4 são conectados aos parafusos ao longo do tanque e L1-L4 são conectados aos pinos de entrada digital do microcontrolador. S0 é conectado a referência 5V.

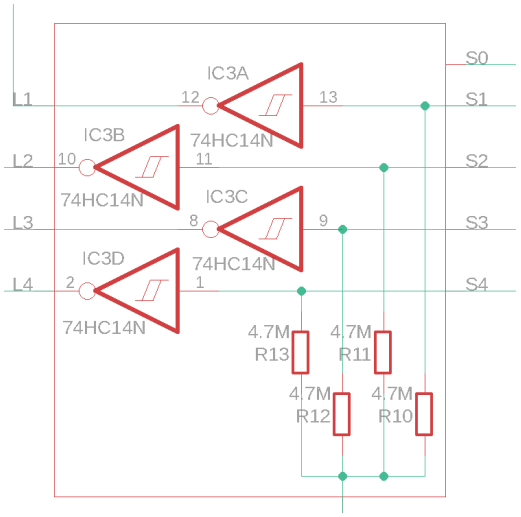


Figure 7: Circuito inversor dos sensores de nível

A quantidade de sensores em nível lógico zero é diretamente proporcional ao nível do tanque e é convertido em uma porcentagem de acordo com um dos seguintes: baseado na porcentagem equivalente da boia, baseado na quantidade de pinos ativos, baseado na altura dos pinos em relação a profundidade total do tanque. Os pinos dos sensores de nível são inicializados como pinos de entrada e é utilizada a função apresentada na figura 8 para contar quantos pinos do tanque estão submersos por água e poder fazer as devidas conversões de porcentagem.

```
/**
 * Count the amount of pins that are under the water
 */
uint16_t get_pins_amount(void)
{
    uint16_t pinsAmount = 0;

    // Check if Pin is low (with water)
    if (!(LEVEL_SENSOR_PIN & (1 << LEVEL_SENSOR_1)))
        pinsAmount++;
    if (!(LEVEL_SENSOR_PIN & (1 << LEVEL_SENSOR_2)))
        pinsAmount++;
    if (!(LEVEL_SENSOR_PIN & (1 << LEVEL_SENSOR_3)))
        pinsAmount++;
    if (!(LEVEL_SENSOR_PIN & (1 << LEVEL_SENSOR_4)))
        pinsAmount++;
    return pinsAmount;
}
```

Figure 8: Contagem de pinos submersos por água

3.3 Modo de controle

O pino INT0 é configurado para servir como uma interrupção externa via botão e é utilizado para selecionar o modo de controle do tanque, o modo selecionado definirá qual a porcentagem do tanque que será utilizada para fazer o controle e para ser exibido para o usuário. O procedimento de inicialização desse pino e a rotina de interrupção podem ser visualizados na figura 9.

```
/**
 * Initialize interrupt at Control selector Pin
 */
void control_selector_init(void)
{
  /* Sensor selector as input with internal pull-up */
  CONTROL_SELECTOR_DDR &= ~(1 << CONTROL_SELECTOR); // Write 0 in desired DDR register
  CONTROL_SELECTOR_PORT |= (1 << CONTROL_SELECTOR); // Write 1 in desired Port register

  /* Interrupt request at falling edge for INT0(PD2) */
  EICRA |= (1 << ISC01);

  /* Enable INT0 */
  EIMSK |= (1 << INT0);

  /* Enable Interrupts */
  sei();
}

/* Interrupt Service Routine for INT0 */
ISR(INT0_vect)
{
  button_count++;
}
```

Figure 9: Inicialização de interrupção externa por botão e rotina de interrupção

Os modos de operação são: 1. controle com a porcentagem da boia; 2. controle com a porcentagem dos sensores de nível; e 3. controle com a média entre a porcentagem da boia e dos sensores de nível.

Estes modos são selecionados através do resultado do resto da divisão entre a quantidade de vezes que a rotina de interrupção foi executada (quantidade de vezes que o botão foi pressionado) e a quantidade de modos existentes no sistema (atualmente 3 modos).

3.4 Filtragem da porcentagem

Após selecionar uma das porcentagens, é aplicado um filtro no qual este valor de porcentagem deve se repetir durante uma quantidade de ciclos de programa definida (neste projeto 15 ciclos) para que o seu valor seja considerado válido para ser exibido ou controlar os atuadores do sistema. Esta solução serve para suavizar os valores apresentados em medidas de sensores intermédios (flutuante).

Os passos do ciclo do programa que executam este filtro podem ser vistos na figura 10 na qual a variável *filter_count* armazena a quantidade de ciclos que o valor de porcentagem se manteve igual ao ciclo anterior, *tank_percent_from_previous_cycle* armazena o valor de porcentagem do ciclo anterior para efeitos de comparação e finalmente *tank_percent_filtered* armazena o valor que será utilizado como entrada nas máquinas de estado dos atuadores e que será exibido pelo display.

```
/* Filtering percent: Change display only if the percent value stay
equals for FILTER_CYCLE_AMOUNT programs cycles */
if (tank_percent_from_previous_cycle != tank_percent_selected)
  filter_count = 0;
else
  filter_count++;

tank_percent_from_previous_cycle = tank_percent_selected;

if (tank_percent_filtered == -1 || filter_count > FILTER_CYCLE_AMOUNT)
  tank_percent_filtered = tank_percent_selected;
/* End of filtering */
```

Figure 10: Passos de filtragem da porcentagem no ciclo do programa

O valor de 15 ciclos foi escolhido empiricamente e apresentou-se como um bom valor para suavizar os valores quando existe uma oscilação muito abrupta dos valores de porcentagem ao longo do tempo.

Uma consideração levantada durante a demonstração do projeto é que, caso a filtragem fosse feita utilizando a média entre as ultimas N amostras de porcentagem, o melhor valor para N neste caso seria 16, pois, por ser uma potência de 2, permite facilmente fazer operações de divisão por este número ao fazer um deslocamento para a direita quatro vezes do numerador em binário.

3.5 Visualização da porcentagem

Neste projeto existem duas formas de visualização da porcentagem do nível do tanque. Uma utilizando dois displays de sete segmentos e outra utilizando uma fileira contendo 8 leds.

Se cada um dos leds fosse controlado por uma porta digital do microcontrolador seriam necessárias $7 * 2 + 8 = 22$ portas apenas para a visualização da porcentagem, o que é inviável por limitações físicas do microcontrolador. Para tornar essa visualização possível foi então utilizado um registrador de deslocamento (Shift Register *74HC595N*) em conjunto com 3 transistores NPN *BC547* reduzindo a quantidade de portas necessárias para obter o mesmo resultado para apenas 6. O circuito esquematico dessa implementação pode ser visualizado na figura 11.

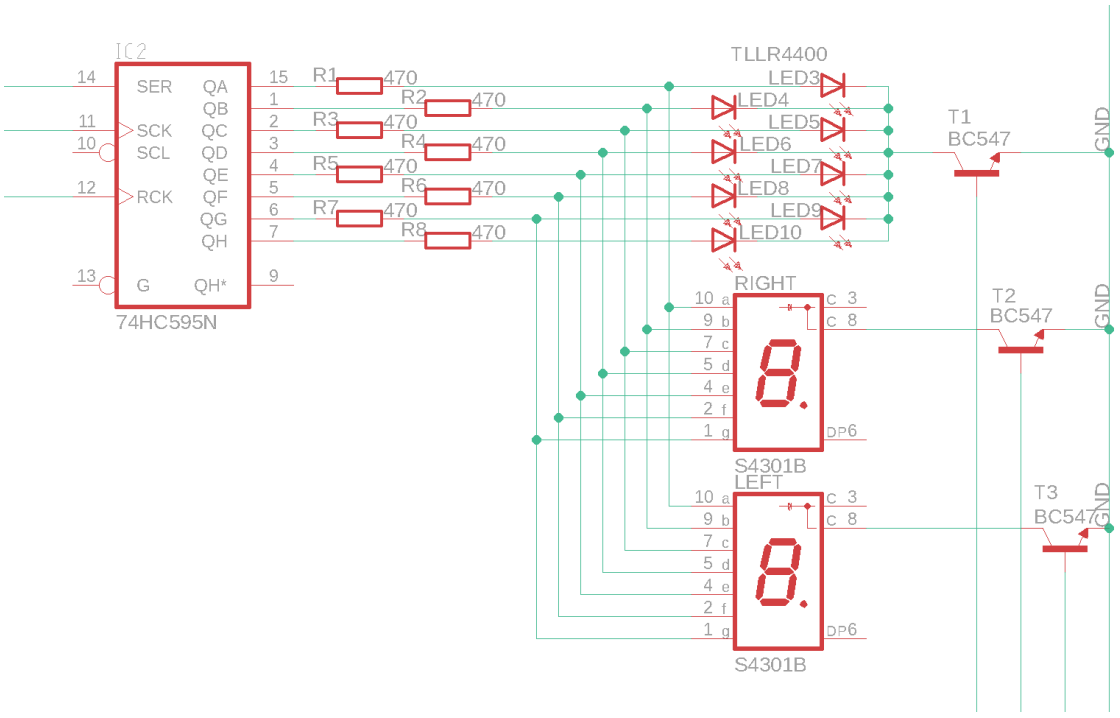


Figure 11: Circuito de controle das formas de visualização da porcentagem

O procedimento controlador do registrador de deslocamento pode ser visualizado na figura 12, no qual o byte passado como parâmetro é percorrido e tem seu valor de cada um de seus bits disponibilizado na pino *SER* enquanto um pulso no pino *SCK* indica que o bit já está disponível na entrada para fazer o deslocamento. Esse processo é repetido até percorrer os 8 bits e apenas quando um pulso é aplicado ao pino *RCK* o valor do byte é disponibilizado na saída do *74HC595N*.

```
/**
 * Provides the byte passed as parameter
 * to the output of the shift register
 */
void shift_register_output(unsigned char byte)
{
  SH_CP_low();
  ST_CP_low();
  for (int i = 0; i < 8; i++)
  {
    if ((byte & (1 << i)))
      DS_high();
    else
      DS_low();
    SH_CP_high();
    SH_CP_low();
  }
  ST_CP_high();
}
```

Figure 12: Procedimento de controle do registrador de deslocamento

Os transistores *BC547* tem suas bases conectadas a três pinos de seleção que são comutados ordenadamente em um intervalo de 5 milisegundos para selecionar qual display apresentará os valores da saída do Shift Register no momento. Devido a persistência retiniana do olho humano esse esse intervalo de 5 milisegundos é suficiente para não se veja os leds piscarem. Este procedimento é apresentado na figura 13.

```c
/*
 * Multiplex between the two seven segments displays and the leds vector
 */
void show_displays(unsigned char percent)
{
    leds_bar_output(percent);
    CLEAR_ALL_DISPLAYS();
    SHOW_LEDS_BAR();
    _delay_ms(5);
    seven_left(percent / 10);
    CLEAR_ALL_DISPLAYS();
    SHOW_LEFT();
    _delay_ms(5);
    seven_right(percent % 10);
    CLEAR_ALL_DISPLAYS();
    SHOW_RIGHT();
    _delay_ms(5);
}
```

Figure 13: Procedimento de comutação entre os três displays

3.6 Acionamento da bomba e alarme

Ambos atuadores, bomba e alarme possuem suas máquinas de estado própias que recebem como entrada o nível do tanque preenchido (em porcentagem) e, de acordo com os limiares definidos, define o estado de acionamento dos atuadores. Os limiares são apresentados na figura 14.

```c
// Define Threshold to activate and desactivate the Motor
#define LOWER_LIMIAR_MOTOR 10  // Motor activate if tank percentage is under this value
#define HIGHER_LIMIAR_MOTOR 50 // Motor desactivate if tank percentage is above this value

// Define Buzzer Parameters
#define UPPER_THRESHOLD_PERCENT 80
#define LOWER_THRESHOLD_PERCENT 20
```

Figure 14: Limiares de ativação da bomba de água e do buzzer

Para evitar níveis de corrente indesejados nas portas do microcontrolador foi contruída uma interface isolada utilizando o *4N25* e um Mosfet *BUZ11* que pode ser visualizado na figura 15. Além disso, foi adicionado um diodo em paralelo com a bomba pois a bomba é uma carga indutiva e quando desligada pode apresentar uma alta tensão inversa e esse diodo serve para proteger o Mosfet de acionamento.

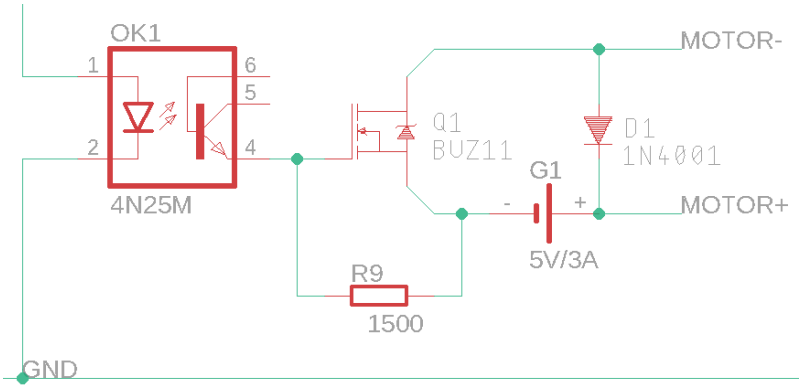


Figure 15: Interface isolada para acionamento da bomba

4 Resultados

O projeto final e circuito prototipado podem ser vistos respectivamente nas figuras 16 e17.

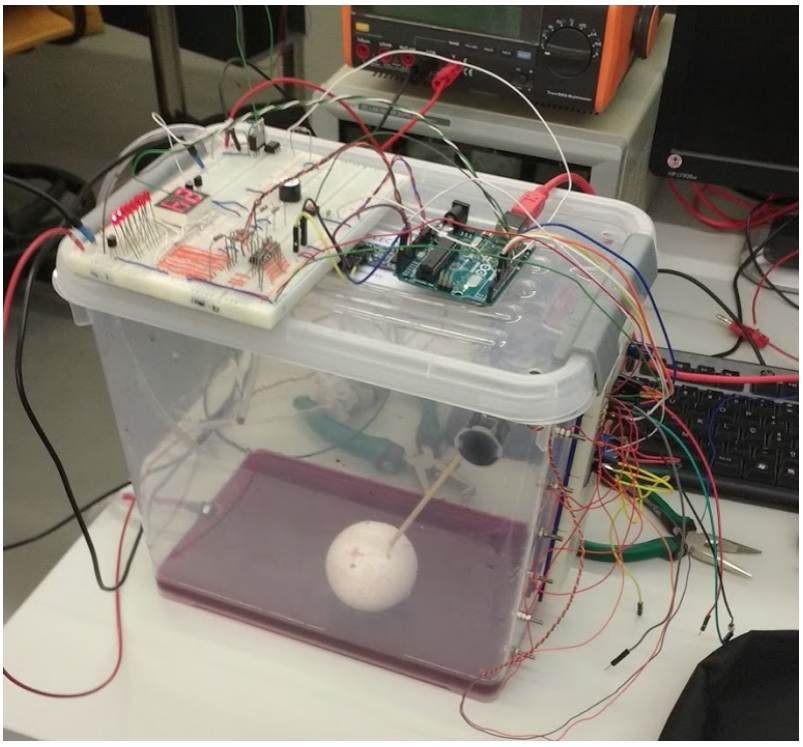


Figure 16: Projeto final do tanque

Na figura 17 pode-se ver o circuito em funcionamento no modo de controle utilizando pinos baseados na quantidade de pinos ativos no qual está sendo representado uma porcentagem de 24% pelos displays de sete segmentos e $\frac{2}{8}$ dos leds do vetor de leds ativos (cerca de 25%).

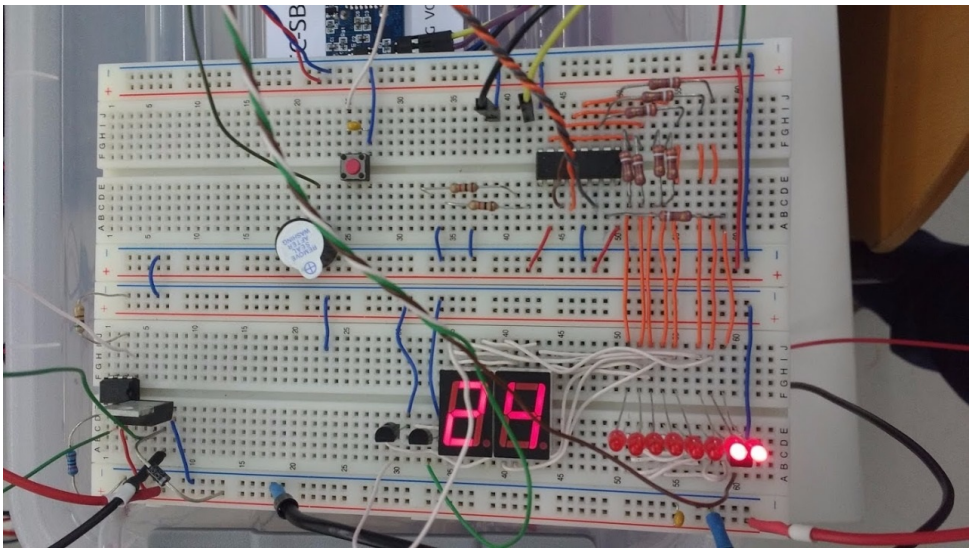
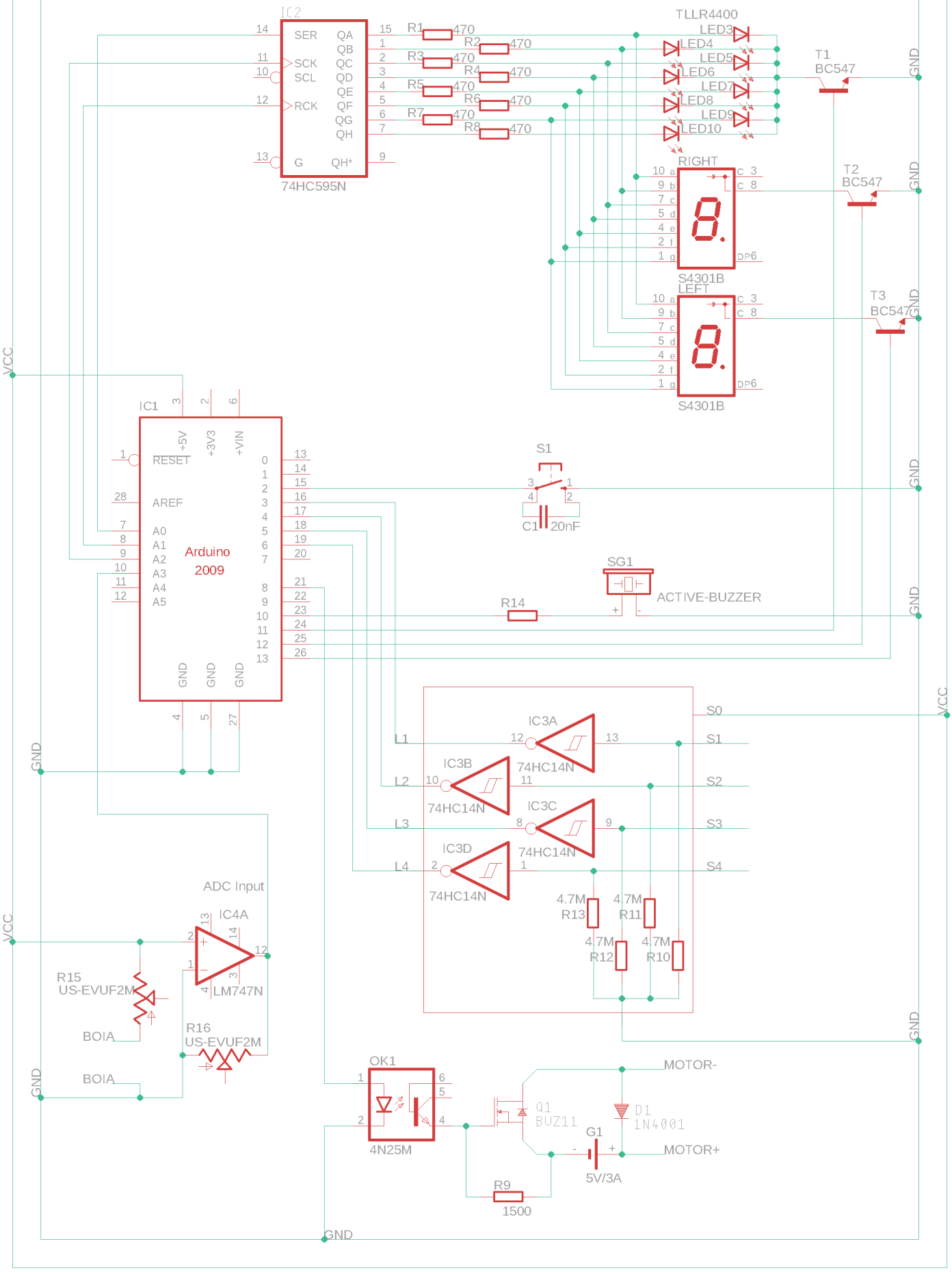


Figure 17: Circuito prototipado

References

[1] The adc of the avr - analog to digital conversion. http://maxembedded.com/2011/06/the-adc-of-the-avr/. Accessed: 2018-12-28.



Responsible dept.	Technical Reference	Document type: Circuit Schematic	Document status
U.PORTO Faculdade de Engenharia	Created by: Cássio Santos	Title, Supplementary title: Tank Fill Control	
	Approved by	projetofinal	Rev. / Date of issue: 28/12/2018 21:38 / Sheet: 1/1