Horta Inteligente – Amigo da Horta 2000

Sistema de Controle e Monitoramento de Morangueiras

Gabriela Barbosa Silva e Pedro Augusto Ferreira da Silva
Faculdade Gama – Engenharia Eletrônica
Universidade de Brasília – UnB
Gama, Brasil
bsgabieng@gmail.com / pedrin.22augusto@gmail.com

***Resumo*—Esse relatório inicial, contém a proposta de projeto final, para a disciplina de Microprocessadores e Microcontroladores. Nesse documento é feito uma explicação do funcionamento do circuito utilizado para a implementação do controlador de umidade e medidor de temperatura, justifica a escolha desse projeto, mostrando a importância do equipamento e propõe os itens que serão entregues ao final do curso.**

Palavras-chave—automação, umidade, monitoramento, morangueira.

I. INTRODUÇÃO

Durante a Segunda Guerra Mundial se deu início à produção e difusão de compostos químicos, com propriedades antibióticas ou inseticidas [1]. Com o crescimento dos centros urbanos, houve um aumento na demanda e produção de produtos agrícolas, aumentando a utilização de compostos químicos para acelerar a produção, isso tem gerado grande impacto no meio ambiente. No Brasil, por exemplo, isso pode ser evidenciado facilmente pela comprovação de grandes quantidades de passivos ambientais registrados atualmente [2].

O Ministério da Saúde estima que mais de 400.000 pessoas são contaminadas anualmente por agrotóxicos, no país e, no mundo, o número de pessoas expostas a estes agentes chega a casa dos milhões [3].

Diante de tantos agravantes desencadeados pelo uso indiscriminado de produtos químicos nos alimentos, faz-se necessário uma conscientização por parte da sociedade e um incentivo à agricultura familiar. Dessa forma, é proposto um sistema que automatize o cultivo de hortaliças em residências, para estimular o cultivo consciente de alimentos, especificamente para a morangueira.

A morangueira necessita de condições específicas para que a mesma dê frutos, como temperatura adequada, não maior do que 30 ºC [4], solo úmido e rico em matéria orgânica [5]. A não observância desses requisitos acarretam em alguns problemas para a morangueira, conforme indicado na figura 1.

Doenças	Condições favoráveis e sobrevivência
Manchas foliares	
Mancha de micosferela *Mycosphaerella fragariae*	Alta umidade, temperatura 20 ~ 25ºC Sobrevivência em restos de cultura
Mancha de dendrofoma **Phomopsis obslurans**	Alta umidade, temperatura elevada ~ 28ºC Sobrevivência em restos de cultura
Mancha de diplocarpon *Diplocarpon earlianum*	Alta umidade, temperatura 20 ~ 25ºC **Sobrevivência em restos de lultura**
Mancha angular *Xanthomonas fragariae*	Alta umidade, temperatura amena ~ 20ºC Sobrevivência em restos de cultura
Oídio *Sphaerotheca macularis* f.sp. *fragariae*	Baixa umidade, temperatura 20 ~ 25ºC Sobrevivência em plantas vivas
Murcha, podridão de rizoma e frutos	
Murcha de verticílio *Verticillium dahliae*	Baixa umidade do solo Após a primeira colheita Sobrevivência em restos de cultura e hospedeiros
Podridão de fitófitora *Phytophthora cactorum*	Alta umidade do solo, temperatura 15 ~ 22ºC Sobrevivência em restos de cultura
Mofo cinzento *Botrytis cinerea*	Alta umidade, temperatura 15 ~ 25ºC Sobrevivência em restos de cultura, plantas hospedeiras
Antracnose	
Podridão de rizoma *Colletotrichum fragariae*	Alta umidade, temperatura ótima ~ 28ºC Sobrevivência em restos de cultura
Flor-preta *Colletotrichum acutatum*	Alta umidade, temperatura ótima ~ 25ºC Sobrevivência em restos de cultura

Figura 1. Principais doenças e condições favoráveis para sua ocorrência [6].

A partir dessas informações, decidiu-se monitorar os dados de umidade do solo, temperatura e luminosidade e controlar a vasão de água para irrigação.

II. REVISÃO BIBLIOGRÁFICA

Outras soluções de automação de hortas ou jardins já existem no mercado. Por exemplo, uma startup de Hong Kong, chamada Aspara, criou um pequeno jardim portátil, para hortaliças, com irrigação automática, o jardim tem um recipiente lateral e, a partir disso, a irrigação é feita, controlada pelo sistema, que distribui a água para os vasos do jardim, o mini jardim também fornece, através de luzes de LED, a luminosidade ideal para o crescimento de cada vegetal, essas luzes alteram sua intensidade de acordo com o estágio crescimento de cada planta, além disso, fatores como temperatura do ambiente, umidade e iluminação, são monitorados e adaptados para o cultivo de diferentes espécies de plantações. O usuário pode monitorar todas essas informações via aplicativo de celular, onde pode acompanhar

quantos dias faltam para a colheita e retardar ou adiantar o processo de crescimento da plantação [7].

Outro pequeno jardim, criado no Colorado, pela empresa ēdn, *Small Garden* 2, vem em um suporte de madeira com alça em alumínio, a caixa compartimentada vem com vários substratos em pequenas porções, onde são colocadas as sementes. O *Small Garden* tem módulo Wi-Fi e possibilita que o usuário consulte informações sobre espécies de planta por um aplicativo, além de monitorar seu pequeno jardim. Pelo aplicativo, também é possível ver o nível de água no reservatório, a umidade do ambiente, acender e apagar a luz do sistema, entre outras funções. O sistema está disponível tanto para o cultivo de pequenas hortaliças, quanto para o cultivo de plantas ornamentais [8].

A Platário, apresentam produtos que já se encontram à venda, nesses produtos, as mudas ficam em vasos com substrato protegidas por uma porta climatizada e a irrigação é automática e feita por subirrigação e capilaridade, nesse caso, o produto tanto pode ser conectado à tubulação da casa, para um resultado 100% automatizado ou pode-se lançar mão do uso da gaveta-reservatório que tem capacidade de água para 10 dias [9].

Com base nas pesquisas de mercado, foi elaborada uma matriz de análise SWOT, para verificar se a escolha do produto era viável, conforme Tabela 1.

	Pontos Fortes	Pontos Fracos
Interno	- Qualidade do produto desenvolvido; - Baixo custo; - Equipe com conhecimentos em eletrônica e software.	- Poucas funcionalidades.
	Oportunidades	**Ameaças**
Externo	- Baixo custo; - Mercado em expansão.	- Recessão econômica.

Tabela 1. Análise SWOT.

Abaixo estão relacionados, os componentes do protótipo e o custo final.

Material	Quantidade	Preço
MSP 430 G2553	1	R$ 80,00
Sensor DHT11	1	R$ 8,00
Sensor de Umidade	1	R$ 15,00
Válvula Solenóide 12 V	1	RS 15,00
Módulo Rele 5 V	1	R$ 15,00
Display Nokia	1	R$ 15,00
Componentes de estrutura	-	R$ 30,00
Valor Total		**R$ 178,00**

Tabela 2. Custo de Projeto

III. Objetivos

O objetivo do projeto é irrigar de forma controlada uma pequena horta de morango, controlando a umidade do solo. Será possível ao usuário monitorar a temperatura do ambiente através de um display.

IV. Planejamento de Projeto

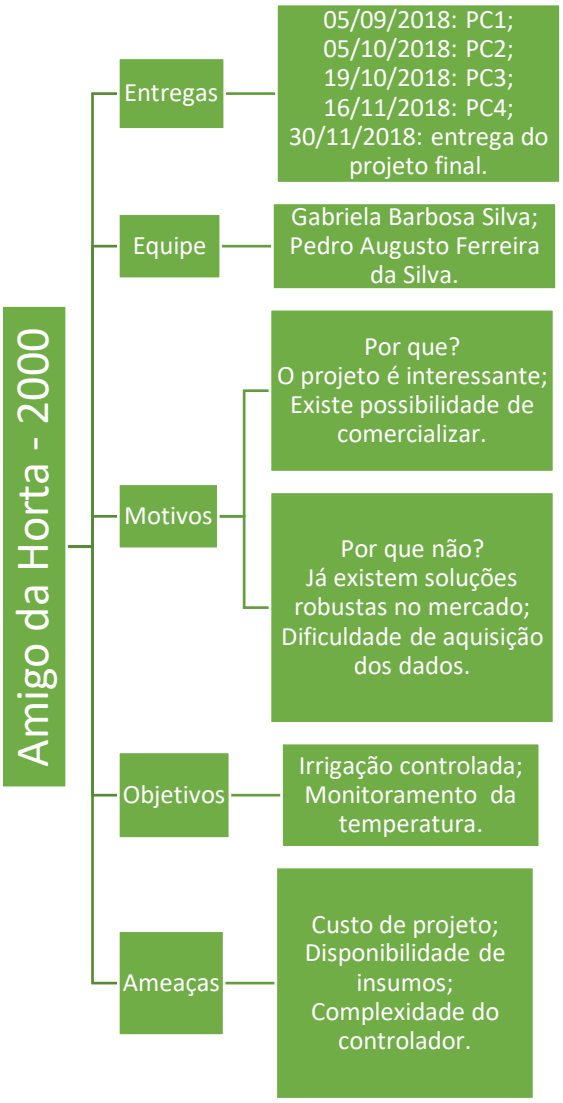


Fig. 1. Planejamento.

V. Requisitos

Será projetado um circuito, que fará a aquisição dos dados de temperatura e umidade do solo, a partir de sensores, após a aquisição, será utilizado o microprocessador, MSP 430, para processar os dados coletados e acionar a liberação controlada de água ao solo, mantendo-o suficientemente úmido. A temperatura será informada ao usuário através de um display, caso esteja acima de 28 °C, será acionado um barulho de alerta.

Fig. 1. Diagrama Funcional.

A. Descrição de Hardware

O Funcionamento do circuito está descrito no fluxograma da Fig. 3. Basicamente será dividido em dois blocos.

Bloco de Controle: O nível de umidade do solo será medido a partir de um sensor de umidade, que, de acordo com a umidade, envia um valor de tensão, quanto menor a umidade, maior a tensão enviada pelo módulo do sensor.

O sensor pode ser alimentado com 3,3 V e é colocado no solo. A partir da resistência do solo, é enviada uma tensão proporcional ao controlador, logo, um solo úmido, apresenta uma baixa resistência, o que gera uma tensão menor. Quando o sistema identifica o solo como seco, tensão menor que 1,4 V, o sistema aciona o relé, que é ativo em nível baixo. O valor enviado pelo sensor é recebido no canal 1 do conversor ADC, de 10 bits, da MSP 430, ver imagem 6. Fazendo-se um cálculo simples de conversão AD, eq. 1, encontra-se o valor digital mínimo para o acionamento da válvula, 480.

$$Resolução = \frac{A_{MAX}}{2^N - 1}$$

Eq. 1

$$Resolução = 2{,}93\ mV/bit$$

A válvula solenoide necessita de 12 V para ser acionada, sendo assim, ela foi ligada a uma fonte que fornece a alimentação apropriada, Fig, 3.

Quando o controlador interpreta um sinal maior que 1,4 V, vindo do sensor, ele leva a porta digital, em que o rele está ligado, para Vss, o que faz com que a válvula solenoide seja aberta por 10 segundos, para evitar que o solo fique alagado.

Bloco de monitoramento: Será utilizado um sensor de temperatura, para que o usuário possa ser alertado caso a temperatura passe de 30 ºC, já que a partir disso, a morangueira não produz frutos, esse dado será indicado em um display Nokia.

O sensor faz a aquisição dos dados de temperatura do ambiente e esses dados são convertidos no canal 10 do ADC da MSP, em seguida, os valores de temperatura são mostrados no display. Quando o valor da temperatura ultrapassa 28 ºC, é enviada uma mensagem de alerta ao display e a MSP leva o pino digital que está ligado no emissor sonoro, dessa forma o usuário é informado sobre o aumento excessivo da temperatura.

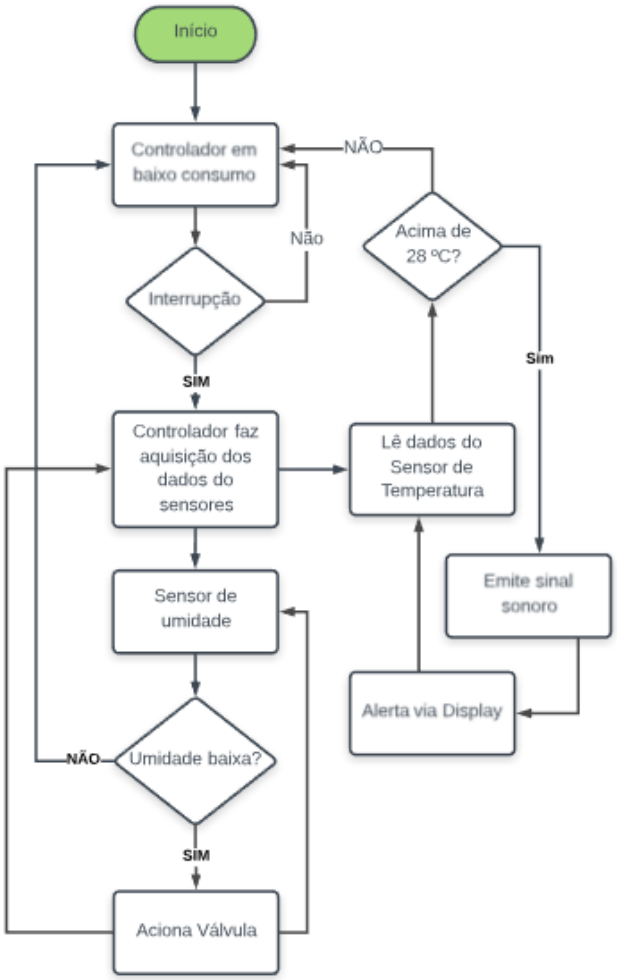


Fig. 2. Fluxograma do Projeto

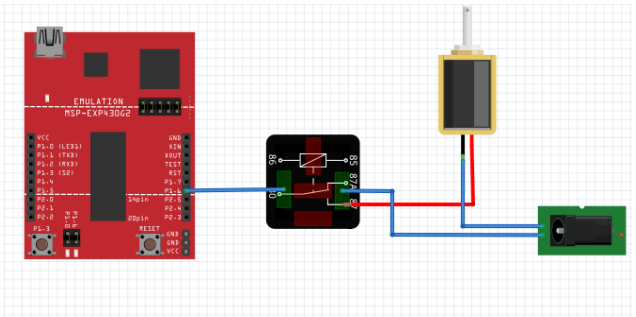
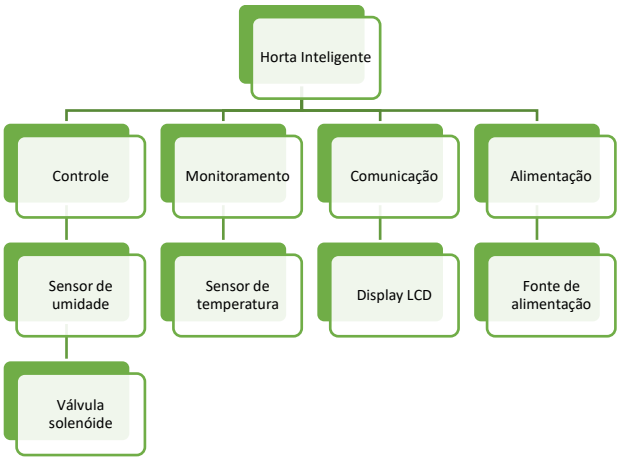


Fig. 3. Bloco de Controle

O diagrama da Fig. 4 ilustra a comunicação dos componentes do sistema.



Fig, 4. Bloco Funcional Atual.

Bloco de Alimentação: O sistema precisa de uma tensão de 12 V para alimentar a válvula solenoide, assim sendo, foi conectada a ela uma fonte de 12 V, que pode ser ligada em uma tomada de 220 Vrms.

B. Descrição de Software

O código do projeto foi realizado no ambiente *Code Composer Studio, CCS*. Para facilitar o funcionamento, debug e construção do código, foram criadas bibliotecas e funções.

```
#include <msp430g2553.h>
#include <legacymsp430.h>

#define IN_AD BIT1
#define IN_AD_CH INCH_1
#define LED1 BIT0
#define LED2 BIT6
#define LEDS (LED1|LED2)
#define Nv 10
```

Fig. 5. Bibliotecas e Variáveis

As variáveis foram definidas através da função “Define”. A partir da Fig. 5, também pode-se perceber que está se utilizando o canal 1 do ADC, para ler os valores de umidade. A Fig. 6 é o trecho do código que utiliza o conversor da MSP.

Na Fig. 7, é possível notar como foi feito para ativar o Timer_A, para que a MSP trabalhasse em modo de baixo consumo, fazendo a leitura dos sensores de tempos em tempos

```
interrupt(ADC10_VECTOR) ADC10_ISR(void)
{
    unsigned int media = 0;
    int cnt;
    v[iv] = ADC10MEM;
    iv++;
    if(iv==Nv)
        iv=0;
    for(cnt=0; cnt<Nv; cnt++)
        media += v[cnt];
    media += Nv/2;
    media /= Nv;
    // Acende ou apaga o LED1
    // de acordo com a leitura AD
    if(media < 490)
        P1OUT &= ~LED1;
    else
        P1OUT |= LED1;

    // Inverte o LED2 para vermos
    // a temporizacao da chamada
    // a esta interrupcao
    P1OUT ^= LED2;
}
```

Fig. 6. Conversor ADC

```
    // Configura o canal 1 do Timer A em modo de comparacao
    // com periodo de 0,5 segundos
    TACCR0 = 6250-1;
    TACCR1 = TACCR0/2;
    TACCTL1 = OUTMOD_7;
    TACTL = TASSEL_2 | ID_3 | MC_1;

    ADC10CTL0 = SREF_0 + ADC10SHT_0 + ADC10ON + ADC10IE;
    ADC10AE0 = IN_AD;
    // Com SHS_1, uma conversao sera requisitada
    // sempre que o canal 1 do Timer_A terminar sua contagem
    ADC10CTL1 = IN_AD_CH + SHS_1 + ADC10DIV_0 + ADC10SSEL_3 + CONSEQ_2;
    ADC10CTL0 |= ENC;

    _BIS_SR(LPM0_bits+GIE);
    return 0;
}
```

Fig. 7. Interrupções do Sistema

No caso do Bloco de Monitoramento, pode-se observar através da Fig. 8, o código que controla o Display Nokia, para utilizar esse Display, também foi utilizada uma biblioteca.

```
    P1OUT |= LCD5110_SCE_PIN + LCD5110_DC_PIN;
    P1DIR |= LCD5110_SCE_PIN + LCD5110_DC_PIN;

    // Configura o módulo USCI
    P1SEL |= LCD5110_SCLK_PIN + LCD5110_DN_PIN;
    P1SEL2 |= LCD5110_SCLK_PIN + LCD5110_DN_PIN;

    UCB0CTL0 |= UCCKPH + UCMSB + UCMST + UCSYNC;    // 3-pin, 8-bit SPI master
    UCB0CTL1 |= UCSSEL_2;                           // SMCLK
    UCB0BR0 |= 0x01;                                // 1:1
    UCB0BR1 = 0;
    UCB0CTL1 &= ~UCSWRST;                           // Apaga SW

    __delay_cycles(500000);
    initLCD();          // Inicia LCD
    clearLCD();         // Apaga o LCD
    tempInit();         // Configura o AD
    writeStringToLCD(" Amigo da horta 2000 ");
    writeStringToLCD("--------------");
    writeStringToLCD("Eletrônica Embarcada 2-2018 ");
    setAddr(24, 5);
    writeGraphicToLCD(setas, NONE);
    writeGraphicToLCD(setas, FLIP_H);
    writeGraphicToLCD(setas, ROTATE);
    writeGraphicToLCD(setas, ROTATE_90_CCW);
```

Figure 8. Display

VI. ETAPA EXPERIMENTAL

A. *Metodologia*

Os testes no sistema foram feitos em cada bloco funcional, separadamente e, em seguida, os códigos foram incorporados a um só programa e os testes foram feitos no sistema completo.

Inicialmente obteve-se muitas dificuldades tanto na definição do escopo de projeto quanto na montagem dos códigos e do circuito, dessa forma, fez-se necessário buscar outra abordagem de projeto. Após pesquisas optou-se por utilizar um método mais detalhado, conforme descrito em [10].

Primeiro foi feito um esboço do circuito, por meio de desenho, para imaginar como ia funcionar e quais funcionalidades de fato eram essenciais e quais eram apenas desejáveis.

Após essa etapa, o comportamento do circuito foi colocado em palavras, foi descrito analiticamente como o circuito deveria funcionar, conforme isso ia sendo feito, foi possível obter mais clareza de como o código deveria funcionar. Nessa etapa foi possível identificar as entradas do circuito e os tipos, duas entradas analógicas umidade e temperatura, logo, foi percebido a necessidade de utilizar um conversor ADC, também foram determinadas, as saídas do projeto, duas saídas digitais, rele e sinal sonoro e uma analógica, display.

Tendo feito isso, o funcionamento do projeto foi quantificado. Nessa etapa foi pesquisado em Data Sheets e feitos experimentos para determinar como os sensores iriam se comportar. A partir de experimentos, foi possível estabelecer a tensão mínima para um solo ser considerado seco e como o sensor de temperatura envia os dados para o controlador. Além disso, foram feitos testes para saber o tempo ideal para a válvula solenoide ficar aberta.

Quando essa etapa foi concluída, foi escrito um esboço de como seria o código e foi percebido as funções que teriam que ser pesquisadas e utilizadas, como conversor ADC e interrupção.

Apenas depois disso foi efetuado o código. Como já citado, os blocos foram testados separadamente, de forma a facilitar a depuração do circuito e do código.

B. *Resultados*

Como explicado anteriormente, foram feitos testes experimentais para definir os valores críticos do projeto, que seriam as condições de laços for, while e de funções do tipo if-else. Sendo assim percebeu-se uma discrepância entre valores indicados nos Data sheets e os valores reais.

No caso do sensor de umidade, a literatura indica que o valor limiar de tensão que indica a umidade do solo é 1,7 V, mas durante as análises experimentais foi possível observar que esse valor era 1,4 V. O experimento foi conduzido através de leituras utilizando o multímetro digital. As Fig. de 9 a 12 ilustram a evolução do projeto.

Fig. 9. Início do Projeto

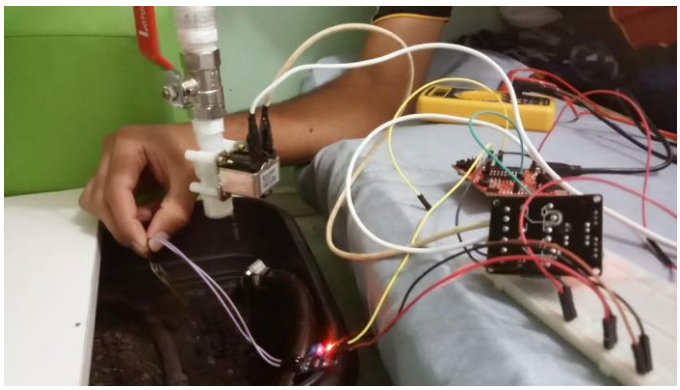


Fig. 10. Testes do Bloco de Controle

Fig. 11. Montagem da Estrutura

Fig. 12. Protótipo Final

Quando o programa dos dois blocos funcionais, foram incorporados à uma main, ocorreram erros. Ao compilar o programa, apenas o sensor de temperatura funcionava. Isso ocorreu porque o programa estava tentando utilizar o mesmo canal do ADC para fazer a conversão.

Conforme a Fig. 13 e 14, pode-se observar o funcionamento do bloco de temperatura que mostra a temperatura ao usuário e, ao detectar uma temperatura maior que 28 ºC, emite sinal sonoro e uma mensagem de alerta.

Figura 13. Sistema informando a Temperatura

Figura 14. Mensagem de alerta

VII. Benefícios

Esse equipamento é utilizado para monitorar e controlar o cultivo de morangos, para incentivar a produção individual de alimentos e para ter uma maior qualidade nos morangos produzidos, diminuindo o risco de doenças. Além de conscientizar o crescimento sustentável a partir de pequenas ações, será possível economia de insumos para o cuidado dos alimentos, já que a água será liberada de forma controlada. Esse equipamento é utilizado para monitorar e controlar o cultivo de vegetais em pequenas hortas, para incentivar a produção individual de alimentos.

VIII. Conclusão

Inicialmente, o projeto não tinha definições mínimas e apresentava funcionalidades que não eram necessárias a essa aplicação, após melhor avaliação e planejamento, foi possível focar nos fatores essenciais e de fato definir uma estrutura. Para compor o código e melhorar o funcionamento do projeto, foi necessário fazer experimento para validar os valores encontrados em literatura.

Além disso, é possível compreender melhor quais utilidades do controlador utilizar, quando se avalia com clareza o objetivo do produto que se está construindo.

IX. Referências

[1] I. M. d. B. S. Stoppelli e C. P. Magalhães, "Saúde e segurança alimentar: a questão dos agrotóxicos," *Ciências e Saúde Coletiva,* pp. 91-100, 2005.

[2] I. C. S. F. Jardim e J. d. A. Andrade, "Resíduos de agrotóxicos em alimentos: uma preocupação ambiental global – um," *Quim. Nova,* vol. 32, nº 4, pp. 996-1012, 2009.

[3] F. PeresI, J. J. Oliveira-Silva, H. V. Della-Rosa e S. R. d. Lucca, "Desafios ao estudo da contaminação humana e ambiental por agrotóxicos," *Ciência e Saúde Coletiva,* vol. 10, pp. 27-37, 2005.

[4] M. J. H. S. I. T. O. M. N. S. A. F. F. R. FRANCINE A SOUSA, "SOMA TÉRMICA DO PLANTIO À COLHEITA PARA O MORANGO EM DIAMANTINA – MG," em *XV Congresso Brasileiro de Agrometeorologia*, Aracaju – SE , 2007.

[5] J. G. C.-P. E. M. R. D. M. D. B. Maria José Alves Bertalot, "Controle alternativo de doenças no morango," Associação Brasileira de Agricultura Biodinâmica.

[6] EMBRAPA, "Palestras," em *2º Simpósio Nacional do Morango,* 2004.

[7] "Aspara.hk," Aspara, [Online]. Available: https://www.aspara.hk/stunning-technology. [Acesso em 03 Outubro 2018].

[8] edn, "edntech," edn, [Online]. Available: https://www.edntech.com/smallgarden. [Acesso em 2018 Outubro 03].

[9] Plantário, “Plantáro,” Plantário, [Online]. Available: https://www.plantario.com.br/. [Acesso em 03 Outubro 2018].

[10] Duke University, *Writing, Running and Fixing Code in C*, 2018.

X. ANEXO

```
#include "msp430g2553.h"

#include "PCD8544.h"

#include <legacymsp430.h>


//********* Declaração de variáveis**********************************************************

char temp[3];

int tempbuff=0;

unsigned char currXAddr = 0; //Variável que armazena a posição atual do cursor em X

unsigned char currYAddr = 0; //Variável que armazena a posição atual do cursor em Y


int iv=0;


//********* Defines**************************************************************************

#define LCD5110_SCLK_PIN BIT5 //Ligar CLK ao pino P1.5

#define LCD5110_DN_PIN BIT7 //Ligar DIN ao pino P1.7

#define LCD5110_SCE_PIN BIT0 //Ligar CE ao pino P1.0

#define LCD5110_DC_PIN BIT1 //Ligar D/C ao pino P1.1

#define LCD5110_SELECT P1OUT &= ~LCD5110_SCE_PIN //LCD5110_SELECT = (0000 0000) & [NOT(0000 0001)] = 0000 000(0)

#define LCD5110_DESELECT P1OUT |= LCD5110_SCE_PIN //LCD5110_DESELECT = (0000 0000) OR (0000 0001) = 0000 000(1)

#define LCD5110_SET_COMMAND P1OUT &= ~LCD5110_DC_PIN //LCD5110_SET_COMMAND = (0000 0000) & [NOT(0000 0010)] = 0000 00(0)0

#define LCD5110_SET_DATA P1OUT |= LCD5110_DC_PIN //LCD5110_SET_DATA = (0000 0000) OR (0000 0010)] = 0000 00(1)0

#define LCD5110_COMMAND 0

#define LCD5110_DATA 1

#define SPI_MSB_FIRST UCB0CTL0 |= UCMSB // USCI B0 Control Register 0 = Async. Mode: MSB first 0:LSB / 1:MSB

#define SPI_LSB_FIRST UCB0CTL0 &= ~UCMSB // USCI B0 Control Register 0 = Async. Mode: LSB first 1:LSB / 0:MSB

#define IN_AD BIT2

#define IN_AD_CH INCH_1

#define LED2 BIT6

#define LEDS (LED2)

#define Nv 10


//********* Configura convesor AD**************************************************************

void tempInit(){

  ADC10CTL0=SREF_1 + REFON + ADC10ON + ADC10SHT_3 ; //3.3V ref,Ref em,64 clocks por amostragem

  ADC10CTL1=INCH_10+ ADC10DIV_3; //temp sensor está em 10 e clock/4

}


//********* Realiza Amostragem e conversão**************************************************************

int tempOut(){

  int t=0;

  __delay_cycles(1000); //Aguarda

  ADC10CTL0 |= ENC + ADC10SC; //Habilita e inicia conversão

  while(ADC10CTL1 & BUSY); //Aguarda enquanto não terminar conversão

  t = ADC10MEM; //salva valor em t

  ADC10CTL0 &=~ENC; //Desabilita a conversão

  return(int) ((t * 27069L - 18169625L) >> 16); //Converte os valores e retorna

}


//********* Protótipos de Funções**********************************************************

void writeStringToLCD(const char *string);
```

```
    void writeCharToLCD(char c);
    void writeBlockToLCD(char *byte, unsigned char length);
    void writeGraphicToLCD(char *byte, unsigned char transform);
    void writeToLCD(unsigned char dataCommand, unsigned char data);
    void clearLCD();
    void clearBank(unsigned char bank);
    void setAddr(unsigned char xAddr, unsigned char yAddr);
    void initLCD();

    //********* Função Principal ************************************************** ******************
    int main() {
        WDTCTL = WDTPW + WDTHOLD;    // Desabilita o WDT
        BCSCTL1 = CALBC1_1MHZ;      // 1MHz clock
        DCOCTL = CALDCO_1MHZ;            // Controle de frequencia em 1MHz

        P1OUT |= LCD5110_SCE_PIN + LCD5110_DC_PIN;
        P1DIR |= LCD5110_SCE_PIN + LCD5110_DC_PIN;

        // Configura o módulo USCI
        P1SEL |= LCD5110_SCLK_PIN + LCD5110_DN_PIN;
        P1SEL2 |= LCD5110_SCLK_PIN + LCD5110_DN_PIN;

        UCB0CTL0 |= UCCKPH + UCMSB + UCMST + UCSYNC;    // 3-pin, 8-bit SPI master
        UCB0CTL1 |= UCSSEL_2;                     // SMCLK
        UCB0BR0 |= 0x01;                          // 1:1
        UCB0BR1 = 0;
        UCB0CTL1 &= ~UCSWRST;                     // Apaga SW

        __delay_cycles(500000);
        initLCD();         // Inicia LCD
        clearLCD();        // Apaga o LCD
        tempInit();        // Configura o AD
        writeStringToLCD("   A H 2000   ");
        writeStringToLCD("--------------");
        writeStringToLCD(" Micro"
              " 2-2018 ");
        setAddr(24, 5);


        // Loop Infinito
        while(1){

            clearBank(4);                 // Apaga a linha 4 do display
            tempbuff = tempOut();         // Faz a Conversão no AD
            temp[0]=((tempbuff/10)+'0');
            temp[1]=((tempbuff%10)+'0');

            writeStringToLCD(" Temperatura:   "
                  "   ");
            writeStringToLCD(temp);
            writeCharToLCD(0x7f);         // Simbolo de graus
            writeStringToLCD("C");

            __delay_cycles(2000000);      // Aguarda 2s
        }
            umidade();


        return 0;
    }

        umidade (void)
    {


        WDTCTL = WDTPW + WDTHOLD;

        BCSCTL1 = CALBC1_1MHZ;
        DCOCTL = CALDCO_1MHZ;

        P1OUT &= ~LEDS;
        P1DIR |= LEDS;
```

```
    // Configura o canal 1 do Timer A em modo de comparacao
    // com periodo de 0,5 segundos
    TACCR0 = 35050-1;
    TACCR1 = TACCR0/2;
    TACCTL1 = OUTMOD_7;
    TACTL = TASSEL_2 | ID_3 | MC_1;

    ADC10CTL0 = SREF_0 + ADC10SHT_0 + ADC10ON + ADC10IE;
    ADC10AE0 = IN_AD;
    // Com SHS_1, uma conversao sera requisitada
    // sempre que o canal 1 do Timer_A terminar sua contagem
    ADC10CTL1 = IN_AD_CH + SHS_1 + ADC10DIV_0 + ADC10SSEL_3 + CONSEQ_2;
    ADC10CTL0 |= ENC;

    _BIS_SR(LPM0_bits+GIE);
    return 0;
}

interrupt(ADC10_VECTOR) ADC10_ISR(void)
{
    unsigned int v[Nv];
    unsigned int media = 0;
    int cnt;
    v[iv] = ADC10MEM;
    iv++;
    if(iv==Nv)
      iv=0;
    for(cnt=0; cnt<Nv; cnt++)
      media += v[cnt];
    media += Nv/2;
    media /= Nv;
    // Acende ou apaga o LED1
    // de acordo com a leitura AD
    if(media < 490)
      P1OUT &= ~LED2;
    else
      P1OUT |= LED2;
}

//********* Função para escrever uma String de caracters no Display ******************************
void writeStringToLCD(const char *string) {
  while(*string) {
    writeCharToLCD(*string++);
  }
}

//********* Função para escrever um caracter no Display ****************************************
void writeCharToLCD(char c) {
  unsigned char i;
  for(i = 0; i < 5; i++) {
    writeToLCD(LCD5110_DATA, font[c - 0x20][i]);
  }
  writeToLCD(LCD5110_DATA, 0);
}

//********* Função para escrever em uma linha inteira do LCD ***************************************
void writeBlockToLCD(char *byte, unsigned char length) {
  unsigned char c = 0;
  while(c < length) {
    writeToLCD(LCD5110_DATA, *byte++);
    c++;
  }
}

//********* Função para desenhar imagens no LCD ************************************************
void writeGraphicToLCD(char *byte, unsigned char transform) {
    int c = 0;
    char block[8];
    if(transform & FLIP_V) {
      SPI_LSB_FIRST;
    }
    if(transform & ROTATE) {
```

```
            c = 1;
            while(c != 0) {
                (*byte & 0x01) ? (block[7] |= c) : (block[7] &= ~c);
                (*byte & 0x02) ? (block[6] |= c) : (block[6] &= ~c);
                (*byte & 0x04) ? (block[5] |= c) : (block[5] &= ~c);
                (*byte & 0x08) ? (block[4] |= c) : (block[4] &= ~c);
                (*byte & 0x10) ? (block[3] |= c) : (block[3] &= ~c);
                (*byte & 0x20) ? (block[2] |= c) : (block[2] &= ~c);
                (*byte & 0x40) ? (block[1] |= c) : (block[1] &= ~c);
                (*byte & 0x80) ? (block[0] |= c) : (block[0] &= ~c);
                *byte++;
                c <<= 1;
            }
        } else {
            while(c < 8) {
                block[c++] = *byte++;
            }
        }

        if(transform & FLIP_H) {
            c = 7;
            while(c > -1) {
                writeToLCD(LCD5110_DATA, block[c--]);
            }
        } else {
            c = 0;
            while(c < 8) {
                writeToLCD(LCD5110_DATA, block[c++]);
            }
        }
        SPI_MSB_FIRST;
    }


    //********* Função para enviar dados ou comando para LCD ******************************************

    void writeToLCD(unsigned char dataCommand, unsigned char data) {
        LCD5110_SELECT;
        if(dataCommand) {
            LCD5110_SET_DATA;
        } else {
            LCD5110_SET_COMMAND;
        }
        UCB0TXBUF = data;
        while(!(IFG2 & UCB0TXIFG));
        LCD5110_DESELECT;
    }


    //********* Função apagar os dados no LCD LCD ***********************************************************

    void clearLCD() {
        setAddr(0, 0);
        int c = 0;
        while(c < PCD8544_MAXBYTES) {
            writeToLCD(LCD5110_DATA, 0);
            c++;
        }
        setAddr(0, 0);
    }


    //********* Função para apagar uma linha inteira do LCD *********************************************

    void clearBank(unsigned char bank) {
        setAddr(0, bank);
        int c = 0;
        while(c < PCD8544_HPIXELS) {
            writeToLCD(LCD5110_DATA, 0);
            c++;
        }
        setAddr(0, bank);
    }


    //********* Função para apontar para um ponto exato no registrador da memoria do display**********

    void setAddr(unsigned char xAddr, unsigned char yAddr) {
        writeToLCD(LCD5110_COMMAND, PCD8544_SETXADDR | xAddr);
        writeToLCD(LCD5110_COMMAND, PCD8544_SETYADDR | yAddr);
```

```
    }


    //********** Função para iniciar o LCD **********************************************************
    void initLCD() {
        writeToLCD(LCD5110_COMMAND, PCD8544_FUNCTIONSET | PCD8544_EXTENDEDINSTRUCTION);
        writeToLCD(LCD5110_COMMAND, PCD8544_SETVOP | 0x3F);
        writeToLCD(LCD5110_COMMAND, PCD8544_SETTEMP | 0x02);
        writeToLCD(LCD5110_COMMAND, PCD8544_SETBIAS | 0x03);
        writeToLCD(LCD5110_COMMAND, PCD8544_FUNCTIONSET);
        writeToLCD(LCD5110_COMMAND, PCD8544_DISPLAYCONTROL | PCD8544_DISPLAYNORMAL);
    }
```