

UNIVERSIDADE FEDERAL DE MATO GROSSO
CAMPUS DE VÁRZEA GRANDE



SMART HOME 2560
Projeto de Sistema Embocado

	Nome	RGA
1	Eduardo Oliveira Silva	201921901002
2	Eduardo Fernandes Bruno dos Reis	202021901012
3	João Victor Villar Boas Pimenta Neves	201921901012
4	Matheo Rodrigues Bonucia	201921901015
5	Pedro Marcos Neves Alves	201921901018

Disciplina: SISTEMAS EMBARCADOS
Prof. Dr. Jésus Franco Bueno

Cuiabá, junho de 2023

SMART HOME 2560
Projeto de Sistema Embarcado

SISTEMA EMBARCADOS

Documentação do Projeto de Sistema Embarcado
apresentado na disciplina de Sistemas Embarcados
como parte da avaliação do aprendizado.

Cuiabá, junho de 2023

Sumário

1 Apresentação do Smart Home 2560	4
2 Modelagem e documentação do Smart Home 2560	5
2.1 Levantamento de Requisitos de Usuário	5
2.2 Definição de Requisitos de Sistema Embocado	6
2.3 Arquitetura do Smart Home 2560	7
2.4 Divisão Entre Componentes de Hardware e Software	8
2.5 Restrições de Projeto do Smart Home 2560	10
3 Definição dos Componentes de Hardware do Smart Home 2560	11
3.1 Microcontrolador ATMEGA2560	11
3.1.1 As portas de conexão do ATMEGA2560	11
3.2 Sensores	13
3.2.1 Sensor de Presença PIR 5011	13
3.2.2 Sensor de Gás MQ-2	13
3.2.3 Sensor de Chuva MD-RD	13
3.2.4 Sensor de temperatura DS18B20	14
3.3 Atuadores	14
4 Montagem do protótipo do Smart Home 2560	15
5 Definição dos componentes de Software	16
5.1 Os módulos de Software	16
5.2 Implementação de Interface de usuários	16
5.3 Código-fonte	18
5.4 Bibliotecas Usadas	26
6 Teste de Campo do protótipo do Smart Home 2560	27
6.1 Teste e Controle das variáveis ambientais pelo Smart Home 2560	27
6.2 Aferição e Calibração dos sensores	27
6.3 Sistema de alimentação de energia	27
7 Referências Bibliográficas	28
8 Apêndice	29

Lista de Figuras

1	Esquemático do projetom ATMEGA2560	7
2	pinMap	12
3	Montagem do protótipo do Smart Home 2560	15
4	Página WEB de usuário	17
5	Principais etapas da definição de um Sistema Embarcado	29

Listings

1	Código Fonte Arduino	18
---	----------------------	----

1 Apresentação do Smart Home 2560

Nos últimos anos houve um interesse grande em automatizar processos dentro de residências. No entanto, é comum que os sistemas de automatização sejam bastante caros. Sendo assim, nossa proposta de projeto consiste em criar um sistema de automatização residencial conectado à rede Ethernet, visando simplificar a vida do usuário e aumentar sua segurança. Esse sistema embarcado foi desenvolvido para residências, principalmente apartamentos, podendo ser instalado em qualquer cômodo.

A principal vantagem desse sistema é seu custo, que é significativamente mais baixo em comparação aos demais sistemas comerciais disponíveis. Ademais, sua montagem é muito simples, facilitando a implementação.

2 Modelagem e documentação do Smart Home 2560

2.1 Levantamento de Requisitos de Usuário

Para definirmos os requisitos de usuário, inicialmente consideramos a criação de um sistema de automação residencial. Sendo assim, precisávamos decidir que tipo de automação seriam implementadas. Após uma análises, identificamos um problema recorrente de acidentes causados por vazamento de gás. Portanto, a primeira medida seria adicionar um sensor de gás que ativesse um alarme quando o nível de gás tivesse fora do normal.

Ademais, outra questão que levamos em consideração é um erro muito comum que várias pessoas comete, esquecer a janelas abertas, molhando toda a casa durante uma chuva inesperada. Para solucionar isso, incluímos um sensor de chuva em nossa lista de requisitos. Na região que moramos, Cuiabá, há uma grande variação de temperatura durante o dia, principalmente na estação de inverno. Sendo assim, achamos útil adicionar dois sensores de temperatura que controlaria a temperatura do ambiente através de sinais infravermelho enviados para o ar condicionado. Um dos sensores seria responsável por passar a informação da temperatura interna e outro para temperatura externa. Dessa forma, caso a temperatura externa estivesse abaixo da temperatura interna, seria enviado um sinal para desligar o ar, recomendando abrir as janelas, e assim economizar energia. Essa implementação também seria útil naqueles momentos que precisamos ligar o ar e não sabemos onde está o controle, pois o processo seria automático.

Além desses itens apresentados, um problema convencional para quem mora sozinho é sair de casa e ficar preocupado se trancou a porta adequadamente, e se alguém poderia entrar escondido. Para criar um solução para esse problema, pensamos em adicionar um sensor de presença infravermelho, que nos alertaria caso fosse ativado enquanto estivéssemos ausentes.

Todas essas informações a partir dos sensores, seriam exibidas num visor LCD fixo. No entanto, pensamos na possibilidade de acessar essas informações fora de casa ou em um cômodo diferente do que está instalado o sistema embarcado. Para isso, incluímos um módulo de Ethernet, que geraria uma página WEB com essas informações, permitindo o controle do sensor de presença e a ativação o desativação do alarme.

Contudo, essas soluções proporcionam maior segurança e conforto aos usuários, abordando problemas comuns enfrentados em residências. Com essa integração dos sensores, o controle e o acesso remoto às informações, nosso sistema de automação residencial visa melhorar a experiência do usuário de forma eficiente.

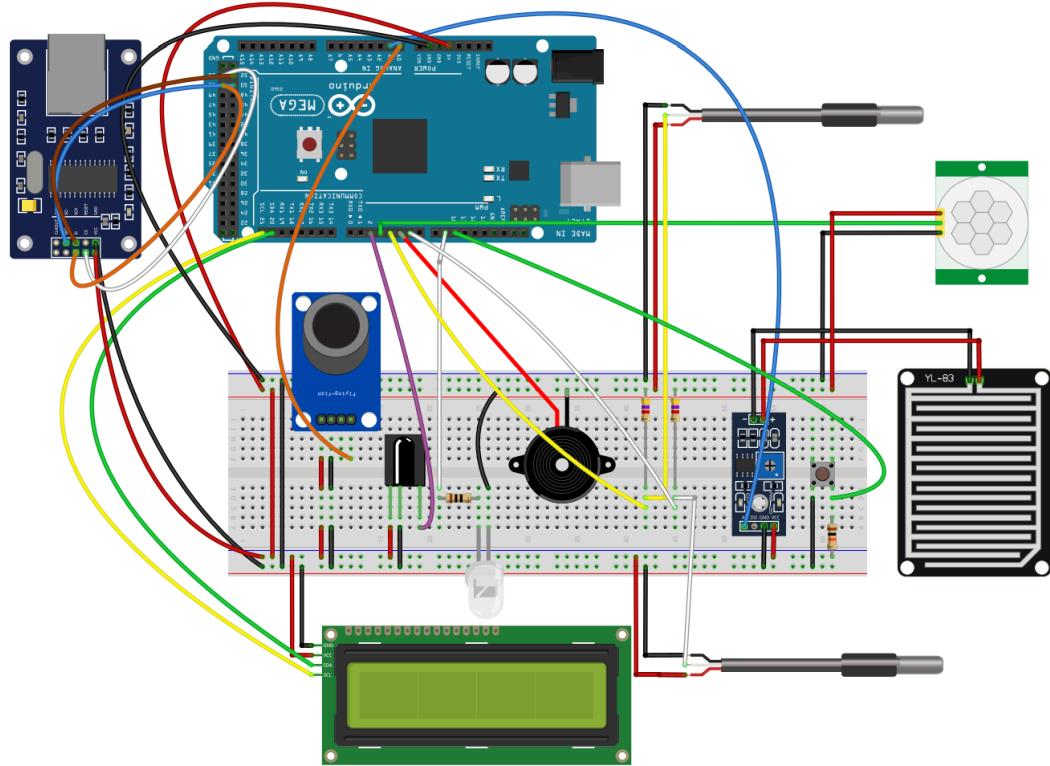
2.2 Definição de Requisitos de Sistema Embarcado

Após levantarmos os requisitos do usuário, partimos para definir os requisitos do Sistema Embarcado. Como mencionado, nosso objetivo era solucionar o problema de vazamento de gás, implementar o controle de temperatura automático, garantir a segurança através do sensor de presença e alertar o usuário quando estiver chovendo com a janela aberta. Nosso principal objetivo era garantir todas essas implementações com o menor custo possível.

Dessa forma, buscamos os módulos com o melhor custo-benefício. Optamos pela plataforma Arduino MEGA, utilizando o microcontrolador ATMEGA2560. No entanto, para reduzir os custos, poderia ser utilizado o Arduino UNO com o ATMEGA328P. Para detectar a chuva, escolhemos o sensor de chuva MH-RD. Já para a detecção de gás, utilizamos o sensor MQ-2, que é capaz de detectar gases inflamáveis e fumaça. Para o alerta, implementamos um buzzer de 5V. Utilizamos um led emissor infravermelho e um receptor infravermelho VS1838B, com uma frequência de 38 kHz, que abrange a maioria dos sinais emitidos por controles de ar-condicionado. Quanto à medição de temperatura, optamos pelo sensor DS18B20, pois sua extensibilidade facilita a instalação em qualquer residência. Para o sistema de segurança, usamos o sensor de presença SR5011. Para exibir as informações ao usuário, utilizamos o LCD 16x20 com módulo I2C. E por fim, para criar um servidor WEB com as informações do sistema embarcado, escolhemos o ENC28J60, por apresentar praticidade e economia.

2.3 Arquitetura do Smart Home 2560

Figura 1: Esquemático do projetom ATMEGA2560



2.4 Divisão Entre Componentes de Hardware e Software

Componentes de hardware utilizamos:

Quantidade	Função	Nome
1	Plataforma de prototipagem	Arduino MEGA
1	Sensor de gás	MQ-2
1	Alarme	Buzzer 5V
1	Visor LCD	LCD I2C 16x2
1	Transmissor infravermelho	LED IR
1	Receptor infravermelho	IR VS1838B
1	Módulo Ethernet	ENC28J60
1	Sensor de Chuva	Rain Drop MH-RD
1	Botão	Button Switch
1	Sensor de presença	PIR HC-SR5011
2	Sensor de temperatura	DS18B20
2	Protoboard	Protoboard 400 furos
2	Resistência sensores temperatura	4.7k Ohm
1	Resistência botão	10k Ohm
1	Resistência LED IR	10 Ohm
10	Fio de conexão	Jumper Fêmea 20cm
35	Fio de conexão	Jumper Macho 20cm

Para componentes de Software utilizamos:

Nome	Função
Biblioteca Wire	Comunicação I2C
Biblioteca LiquidCrystal_I2C	Controlar o módulo LCD
Biblioteca DallasTemperature	Converter e transmitir os sinais do sensor de temperatura
Biblioteca IRremote	Transmitir sinais infravermelho
Biblioteca EtherCard	Controlar o módulo ENC28J60 conectando o Arduino à Ethernet
Biblioteca Arduino	Controlar o Arduino utilizando a linguagem C++
Platformio	Plataforma de desenvolvimento

2.5 Restrições de Projeto do Smart Home 2560

Nosso projeto, apresenta algumas restrições que futuramente pode ser aprimorada para melhorar a usabilidade.

A primeira restrição: é a necessidade de estar conectado a um modem por meio de um cabo Ethernet RJ45, resultando em um fio adicional durante a montagem. Para reduzir a presença de fios e aumentar a flexibilidade, pode ser considerado um módulo Wifi, ou utilizar um Arduino com Wifi embutido.

A segunda restrição: é o fato do módulo sensor de gás está localizado na protoboard próximo ao microcontrolador. Sendo assim, essa configuração pode não ser ideal se a cozinha for muito longe do sistema, pois a coinhza é onde possui o maior risco de vazamento de gás. Sendo assim, para utilizar o sensor na cozinha seria necessário realizara extensão dos cabos.

A terceira restrição: é que nem todos os modems permitem a conexão direta do módulo ENC28J60 à Ethernet, o que pode ocasionar problemas de conexão dependendo da operadora.

A quarta restrição: no projeto atual, para medir a temperatura externa é necessário realizar um furo na parede para passar o sensor.

A quinta restrição: o alcance do emissor infravermelho é limitado, então o sistema precisa ser posicionado de modo que seja compatível com o receptor do ar-condicionado.

Dessa forma, podemos perceber que o projeto possui algumas restrições de uso que podem ser resolvida com pequenas melhorias trocando alguns componentes, entretanto isso geraria um custo maior.

3 Definição dos Componentes de Hardware do Smart Home 2560

3.1 Microcontrolador ATMEGA2560

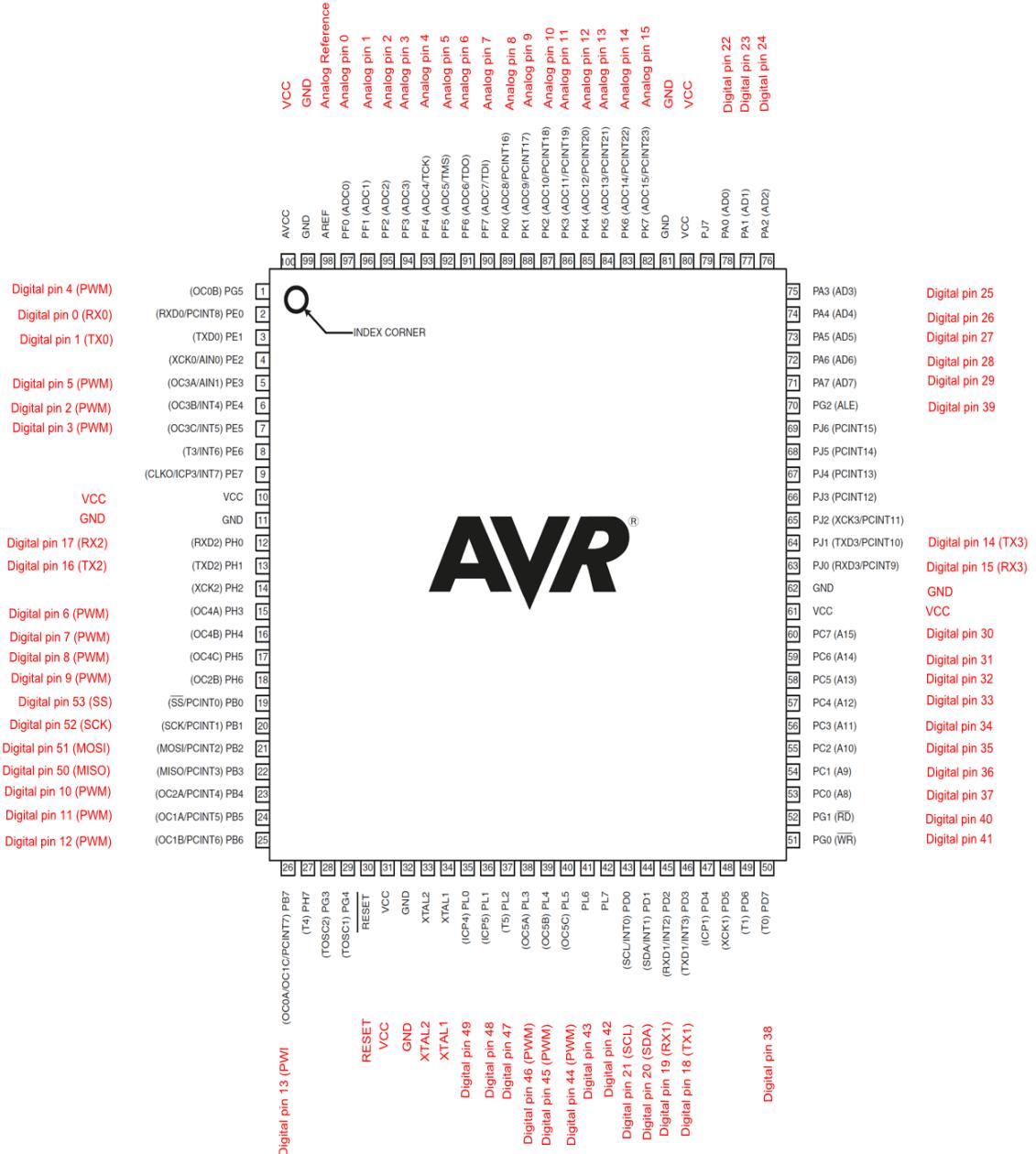
Nesse projeto, utilizamos o microcontrolador ATMEGA2560, embutido no Arduino MEGA. A escolha desse microcontrolador foi baseada nas suas características avançadas e versatilidade.

O ATMEGA2560 é um microcontrolador de alto desempenho, baseado na arquitetura AVR, possui uma variedade de recursos, tornando-o ideal para aplicações complexas e que requer bastante memória. O motivo da sua escolha foi principalmente isso, pois como iamos gerar um servidor com uma página WEB, iria precisar de bastante memória. Possui um capacidade de memória de 256 kb de memória Flash e 8 kb de memória RAM.

3.1.1 As portas de conexão do ATMEGA2560

Porta ATMEGA2560	Porta Arduino MEGA	Categoria	Função
VCC	5V	VCC	Fonte de energia
GND	GND	Ground	Ground
PF0 (ADC0)	A0	Analógica	Sensor de Gás
PF1 (ADC1)	A1	Analógica	Sensor de Chuva
PD1	20	SDA	Comunicação LCD
PD0	21	SCL	Comunicação SCL
PE4	D2	PWM	Receptor Infravermelho
PE5	D3	PWM	Sensor de presença
PG5	D4	PWM	Sensor de temp. externo
PE3	D5	PWM	Buzzer
PH3	D6	PWM	Sensor de temp. interno
PH6	D9	PWM	Emissor Infravermelho
PB4	D10	PWM	Botão Switch
PB3	D50	Digital	MOSO
PB2	D51	Digital	MOSI
PB1	D52	Digital	SCK
PB0	D53	Digital	CS

Figura 2: pinMap



Fonte: docs.arduino.cc (Acesso em 05 de junho de 2023)

3.2 Sensores

3.2.1 Sensor de Presença PIR 5011

É um dispositivo utilizado para detectar a presença de pessoas em um ambiente. Funciona com base na detecção de radiação infravermelha emitida por corpos em movimento. Detecta uma variação de até 100° há uma distância de até 7 metros.

Porta Sensor	Porta Arduino
DOUT	D3
VCC	VCC
GND	GND

3.2.2 Sensor de Gás MQ-2

Esse sensor é projetado para detectar diferentes tipos de gases inflamáveis e fumaça. Utiliza da combinação de sensor de aquecimento e um sensor de gás que mede a concentração de gás em um ambiente.

Porta Sensor	Porta Arduino
A0	A0
D0	
VCC	VCC
GND	GND

3.2.3 Sensor de Chuva MD-RD

Esse sensor permite a detecção de chuva ou umidade. Possui um superfície exporta sensível à água que é capaz de conduzir eletricidade quando entra em contato com umidade.

Porta Sensor	Porta Arduino
A0	A1
D0	
VCC	VCC
GND	GND

3.2.4 Sensor de temperatura DS18B20

É um sensor digital de alta precisão que mede temperatura em um determinado ambiente, possui um fio de extensão e uma ponta de detecção. Possui apenas um fio para transmitir os dados lido. Consegue detectar uma variação de -55° até 125° .

Externo

Porta Sensor	Porta Arduino
VCC	VCC
Data	D4
GND	GND

Interno

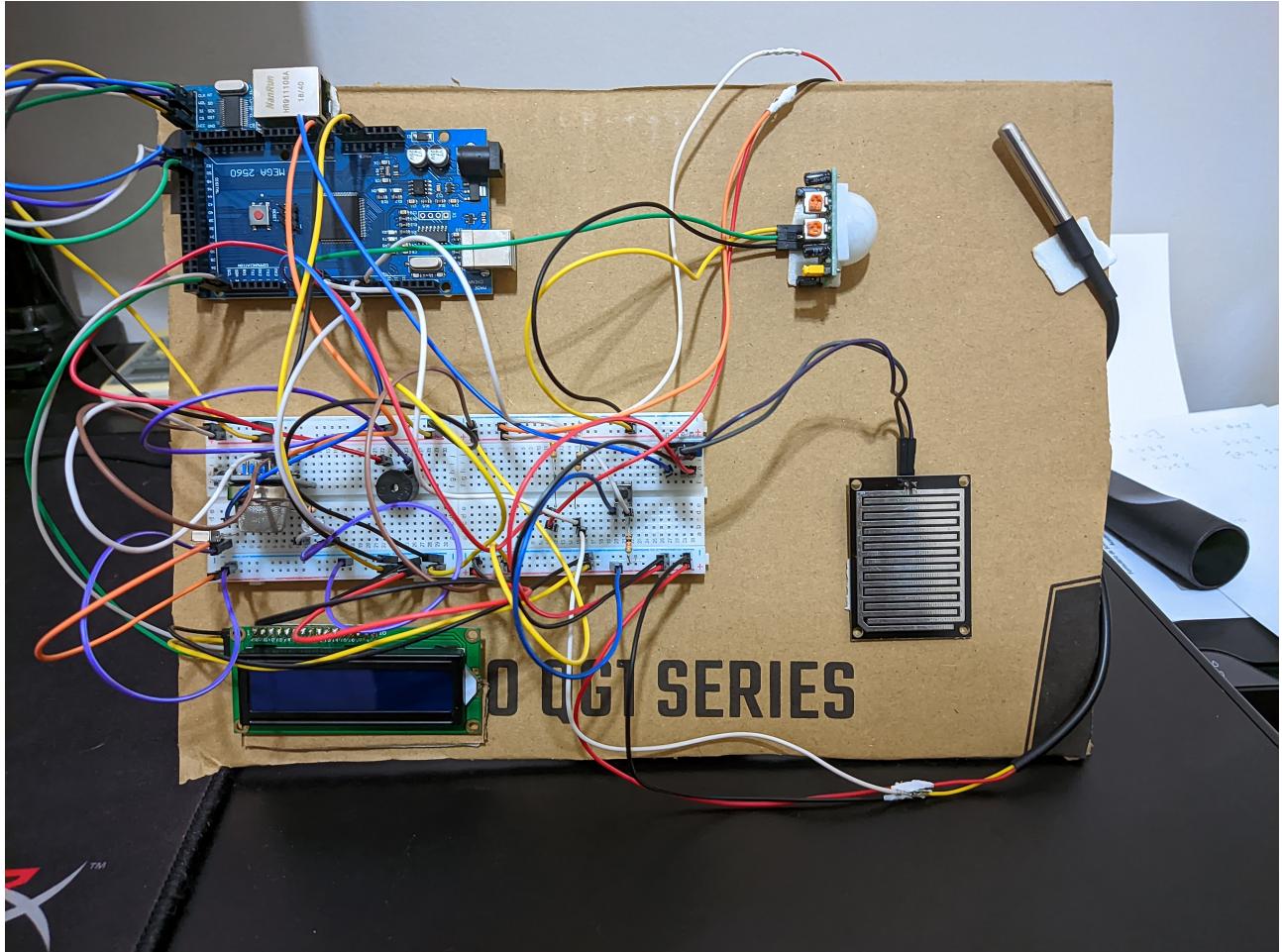
Porta Sensor	Porta Arduino
VCC	VCC
Data	D6
GND	GND

3.3 Atuadores

Como atuadores, temos um buzzer e um LED emissor de infravermelho. O buzzer é acionado quando o detector de presença é ativado ou quando o nível de gás está acima do normal. Já o LED emissor de infravermelho envia um sinal RAW IR para o ar-condicionado quando o sensor de temperatura interna detecta uma temperatura superior a $32^{\circ}C$. Isso faz com que o ar-condicionado seja ligado automaticamente a uma temperatura pré-definida de $24^{\circ}C$, conforme especificado no código RAW.

4 Montagem do protótipo do Smart Home 2560

Figura 3: Montagem do protótipo do Smart Home 2560



5 Definição dos componentes de Software

5.1 Os módulos de Software

O software do nosso sistema possui uma configuração de alarme ativado a partir de um botão. Quando o botão é pressionado, é registrado em uma variável que o botão foi acionado. Em seguida, assim que o sensor de presença é acionado, significa que, que o alarme será acionado.

Além disso, implementamos um alarme para quando o nível de gás estiver acima do normal. Observamos que até 300 é considerado um valor normal. Portanto, assim que a leitura ultrapassar 300, o alarme será acionado.

Implementamos também um função de leitura de sinal infravermelho, que servirá para cadastrar uma nova função para o emissor no futuro. Existe uma variável chamada *RECEPTOR*, que é definida no escopo. Portanto, se eviarmos o firmware para o Arduino com essa variável definida como HIGH, o sistema funcionará apenas para ler o sinal infravermelho recebido e transmiti-lo através do serial a uma taxa de 115200.

Por fim, criamos 3 telas diferentes que cada uma aparece durante 20 segundos. Utilizamos uma técnica de salvar o tempo inicial assim que é realizado o setup e criar uma função de exibição para cada tela, então no loop é atualizada o valor de uma variável referente ao tempo atual, então enquanto a diferença do tempo atual em relação do tempo inicial for menor que o tempo de tela 1, chama a função da tela 1. Dessa forma, é implementada essa lógica em todas as telas, isso é útil pois quanto um tela está sendo exibida, as informações são atualizadas em tempo real.

5.2 Implementação de Interface de usuários

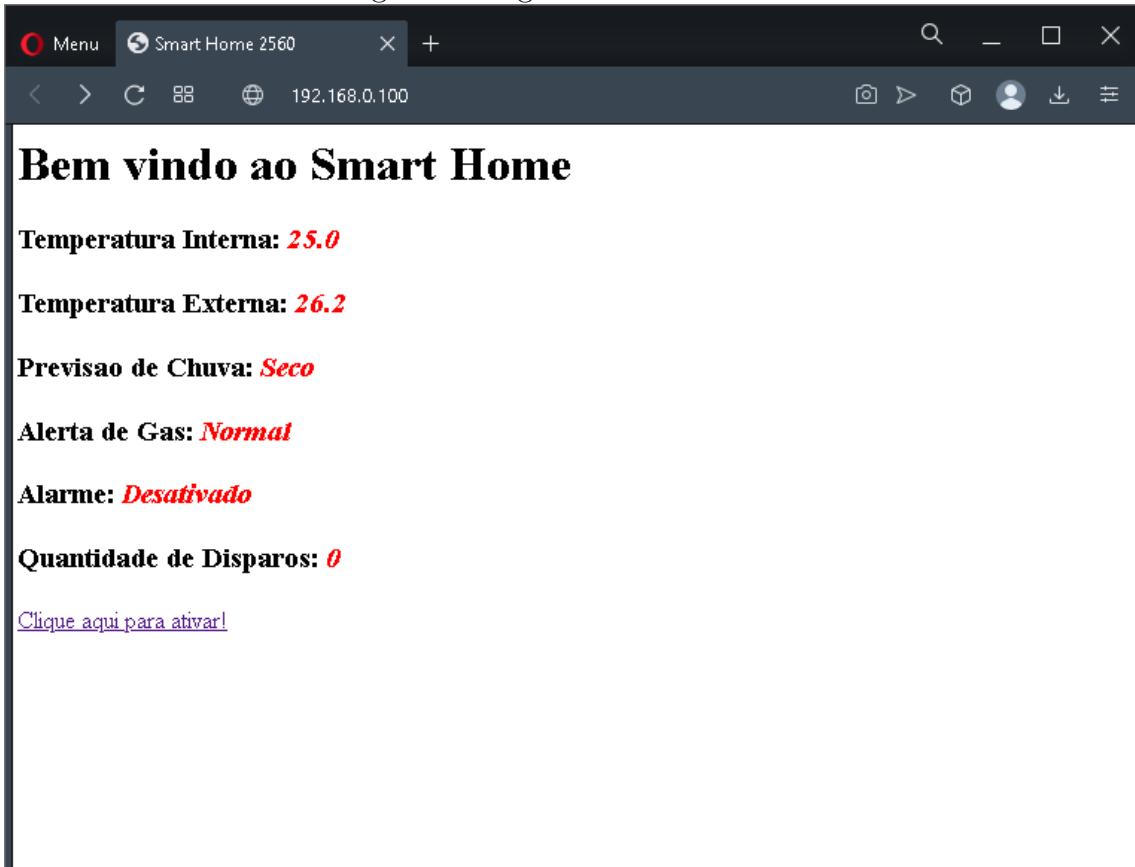
Na interface do usuário, são 3 telas no total. A primeira tela mostra temperatura interna e externa. Se a temperatura estiver acima de $32^{\circ}C$ será enviados 3 sinais com um intervalo de 6 segundos para ligar o ar-condicionado na temperatura de $24^{\circ}C$.

A segunda tela mostra as condições climáticas, indicando se está chovendo muito, garoando ou se está seco. Além disso, exibe o nível de gás, com as seguintes categorias: Alerta(acima de 200 e menor que 300), Perigo (acima de 300) e Normal (abaixo de 200).

Ademais, a terceira tela mostra a quantidade de disparos de alarme que ocorreram e indica se o alarme está ativo ou não. O usuário pode ativá-lo e desativá-lo simplesmente apertando o botão.

Por fim, existe a interface WEB gerada a partir de HTML, utilizando a biblioteca EtherCard é gerado uma página HTML com as informações do sensores a partir de uma requisição GET.

Figura 4: Página WEB de usuário



5.3 Código-fonte

```

1 #include <Wire.h>
2 #include <LiquidCrystal_I2C.h>
3 #include <DallasTemperature.h>
4 #include <IRremote.h>
5 #include <EtherCard.h>
6 IRsend irsend; // instancia o objeto emissor de sinal IR
7
8 LiquidCrystal_I2C lcd(0x27, 16, 2); // instancia o LCD
9
10 #define RECEPTOR 0 // 1 para rebecer sinal IR do controle
11
12 #if RECEPTOR // variaveis e funcoes para recepcao de sinal IR
13 #define maxlen 800
14 #define pinIR 2
15 #define LEDPIN 13
16 volatile unsigned int irBuffer[maxLen];
17 volatile unsigned int x = 0;
18
19 void IR_Interrupt_Handler() {
20     if (x > maxlen) return;
21     irBuffer[x++] = micros();
22
23 }
24 #endif
25
26 #define gas A0
27 #define rain A1
28 #define valorNormal 300
29
30 static byte myip[] = {192, 168, 0, 100}; // endereço IP
31 static byte gwip[] = {192, 168, 0, 1}; // endereço Gateway
32 static byte mymac[] = {0x74, 0x69, 0x69, 0x2D, 0x30, 0x31}; // endereço MAC
33
34
35
36 const int temperatura_ext = 4;
37 const int temperatura_int = 6;
38
39 const int pinPIR = 3;
40 const int buzzer = 5;
41 const int button = 10;
42
43 String clima = "";

```

```

44 String alertaGas = "";
45
46 int tentativas = 0;
47
48
49 unsigned long tempo_inicial;
50 unsigned long tempo_atual;
51 const unsigned long tempo_tela1 = 20000; // 20 segundos
52 const unsigned long tempo_tela2 = 20000;
53 const unsigned long tempo_tela3 = 20000;
54
55 int alarm = 0;
56 int count = 0;
57
58
59 unsigned int S_pwr[349]={4392, 4484, 524, 1628, 604, 484, 604, 1604, 576,
1604, 600, 488, 604, 488, 600, 1556, 624, 488, 604, 488, 576, 1580, 624,
488, 576, 512, 580, 1576, 628, 1556, 600, 512, 576, 1604, 604, 1576, 576,
516, 524, 1656, 524, 1656, 600, 1576, 604, 1580, 600, 1576, 604, 1576,
604, 488, 600, 1556, 624, 488, 600, 492, 600, 488, 600, 492, 600, 492,
572, 516, 600, 488, 604, 1576, 604, 488, 600, 488, 576, 516, 576, 512,
600, 492, 600, 488, 600, 1580, 604, 488, 600, 1580, 600, 1580, 572, 1608,
600, 1580, 572, 1608, 572, 5240, 4492, 4360, 624, 1580, 600,
488, 604, 1576, 604, 1576, 600, 492, 576, 512, 524, 1632, 624, 488, 528,
564, 600, 1580, 600, 492, 524, 564, 576, 1604, 524, 1656, 524, 544, 596,
1608, 600, 1580, 596, 492, 600, 1556, 624, 1580, 524, 1656, 600, 1580,
524, 1656, 600, 1552, 552, 564, 600, 1580, 524, 568, 520, 568, 600, 468,
596, 516, 524, 568, 600, 488, 576, 516, 524, 1656, 596, 492, 572, 520,
600, 488, 576, 516, 572, 516, 600, 492, 600, 1580, 524, 564, 600, 1580,
600, 1580, 600, 1580, 600, 1580, 600, 1580, 600, 1580, 600, 55856, 144};
60 //ligar ar (sinal do ar da marca eletrolux configura a 24 graus)
61
62 int movimento = LOW;
63
64 OneWire oneWireExt(temperatura_ext);
65 // instancia o objeto oneWire para comunicacao com o sensor de temperatura
// externa
66 OneWire oneWireInt(temperatura_int);
67 // instancia o objeto oneWire para comunicacao com o sensor de temperatura
// interna
68
69
70 DallasTemperature sensorExt(&oneWireExt);
71 // instancia o objeto sensor de temperatura externa
72 DallasTemperature sensorInt(&oneWireInt);

```

```

73 // instancia o objeto sensor de temperatura interna
74
75 static word homePage(int chuva, int valorGas, double tempExt, double tempInt
76   , int alrm, int count){ // pagina WEB
77 bfill = ether.tcpOffset();
78 bfill.emit_p(PSTR(
79 "HTTP/1.0 200OK\r\n"
80 "Content-Type: text/html\r\n"
81 "Pragma: no-cache\r\n"
82 "\r\n"
83 "<html><title>"
84   "Smart Home 2560"
85 "</title></head>"
86 "<body>"
87   "<h1>Bem vindo ao Smart Home</h1>"
88   "<h3>Temperatura Interna: <em style='color:red'>"tempInt"</em></h3>"
89   "<h3>Temperatura Externa: <em style='color:red'>"tempExt"</em></h3>"
90   "<h3>Previsao de Chuva: <em style='color:red'>"chuva"</em></h3>"
91   "<h3>Alerta de Gas: <em style='color:red'>"valorGas"</em></h3>"
92   "<h3>Alarme: <em style='color:red'>"alrm"</em></h3>"
93   "<h3>Quantidade de Disparos: <em style='color:red'>"count"</em></h3>"
94   "<p><a href=/almr=1>Clique aqui para ativar o alarme!</a></p>"
95
96   "</body>"
97 "</html>"));
98 return bfill.position();
99 }
100
101 int rainSensor() { // sensor de chuva
102   int valorSensor = analogRead(rain);
103   int valorSaida = map(valorSensor, 0, 1023, 255, 0);
104   return valorSaida;
105 }
106
107 void ligarAr(){ // funcao para ligar o ar
108   Serial.println("Ligando... ");
109   irsend.sendRaw(S_pwr, 349, 38);
110   Serial.println("Executando a 38 Hz");
111   digitalWrite(13, HIGH);
112   delay(100);
113   digitalWrite(13, LOW);
114   tentativas++;
115   delay(3000);
116 }
```

```

117
118 void alarme_sonoro(){ // funcao para disparar o alarme
119     count++;
120     for(int i = 0; i < 20; i++){
121         digitalWrite(buzzer, HIGH);
122         analogWrite(buzzer, 200);
123         delay(100);
124         analogWrite(buzzer, 25);
125         delay(100);
126     }
127     digitalWrite(buzzer,LOW);
128 }
129
130 void atualizaTela1(double tempExt, double tempInt){
131
132     sensorExt.requestTemperatures();
133     sensorInt.requestTemperatures();
134     double tempExt = sensorExt.getTempCByIndex(0);
135     double tempInt = sensorInt.getTempCByIndex(0);
136
137     tempExt = sensorExt.getTempCByIndex(0);
138     tempInt = sensorInt.getTempCByIndex(0);
139
140     lcd.setCursor(0,0);
141     lcd.print("Temp.Int: ");
142     lcd.setCursor(9,0);
143     lcd.print(tempInt);
144     lcd.print(" C");
145
146     lcd.setCursor(0,1);
147     lcd.print("Temp.Ext: ");
148     lcd.setCursor(9,1);
149     lcd.print(tempExt);
150     lcd.print(" C");
151
152     if(tempInt > 32.0 && tentativas <= 3){
153         // se a temperatura interna for maior que 32 graus e o sinal nao tiver
154         // sido enviado mais de 3 vezes
155         ligarAr();
156     }
157 }
158
159 void atualizaTela2(int chuva, int valorGas){
160     chuva = rainSensor();

```

```
161 valorGas = analogRead(gas);
162
163 if(valorGas > 200 && valorGas < 300) {
164     alertaGas = " Alerta      ";
165 } else if(valorGas > 300){
166     alertaGas = " Perigo!    ";
167     alarme_sonoros();
168 } else{
169     alertaGas = " Normal      ";
170 }
171
172 if(chuva >= 100){
173     clima = " Chuva   ";
174 } else if(chuva > 50 && chuva < 100){
175     clima = " Garoa   ";
176 } else{
177     clima = " Seco    ";
178 }
179
180 lcd.setCursor(0,0);
181 lcd.print("Previsao:");
182 lcd.setCursor(9,0);
183 lcd.print(clima);
184
185 lcd.setCursor(0,1);
186 lcd.print("Gas:");
187 lcd.setCursor(4,1);
188 lcd.print(alertaGas);
189
190 }
191
192 void atualizaTela3(){
193     lcd.setCursor(0,0);
194     lcd.print("Alarme:");
195     lcd.setCursor(7,0);
196     if(alrm){
197         lcd.print(" Ativo    ");
198     } else{
199         lcd.print(" Inativo  ");
200     }
201     lcd.setCursor(0,1);
202     lcd.print("Qt.Disparos:");
203     lcd.setCursor(12,1);
204     lcd.print(count);
205 }
```

```

206 }
207
208 void ativaAlarme(int alarme){
209     movimento = digitalRead(pinPIR);
210     if(alarme == HIGH && alrm == 0){
211         alrm = 1;
212     }else if(alarme == HIGH && alrm == 1){
213         alrm = 0;
214     }
215     if(alrm == 1){
216         if(movimento == HIGH){
217             alarme_sonoro();
218         }
219     }
220 }
221
222
223 void setup() {
224     lcd.init();
225     lcd.backlight();
226     lcd.setCursor(0,0);
227     lcd.print("Smart Home :)");
228
229     pinMode(buzzer,OUTPUT);
230     pinMode(pinPIR, INPUT);
231     pinMode(button, INPUT);
232     pinMode(13, OUTPUT);
233
234     delay(5000); // calibrando sensores
235     Serial.begin(115200);
236
237     sensorInt.begin();
238     sensorExt.begin();
239
240     if (ether.begin(sizeof Ethernet::buffer, mymac) == 0) { // inicializa o
241         modulo ethernet com o endereço MAC
242         Serial.println(F("Falha ao comunicar com o Ethernet shield"));
243         lcd.println("Falha Rede");
244     }else{
245         Serial.println(F("Ethernet OK"));
246         lcd.println("Rede OK");
247         ether.staticSetup(myip, gwip); // configura o endereço IP e Gateway
248     }
249

```

```

250
251     tempo_inicial = millis(); // inicializa o tempo inicial
252
253 #if RECEPTOR
254     attachInterrupt(digitalPinToInterrupt(pinIR), IR_Interrupt_Handler, CHANGE
255 );
256 #endif
257 }
258
259 void loop() {
260 #if RECEPTOR // codigo para recepcao de sinal IR do controle remoto em
261     formato RAW exibindo o sinal no monitor serial
262     lcd.setCursor(0,0);
263     lcd.print("Recebendo Sinal");
264
265     Serial.println(F("Press the button on the remote now - once only"));
266     delay(5000);
267     lcd.clear();
268     if (x) {
269         digitalWrite(LEDPIN, HIGH);
270         Serial.println();
271         Serial.print(F("Raw: "));
272         lcd.setCursor(0,1);
273         lcd.print("Sinal Recebido! ");
274         Serial.print((x - 1));
275         Serial.print(F(" "));
276         detachInterrupt(digitalPinToInterrupt(pinIR));
277         for (int i = 1; i < x; i++) {
278             if (!(i & 0x1)) Serial.print(F("-"));
279             Serial.print(irBuffer[i] - irBuffer[i - 1]);
280             Serial.print(F(", "));
281         }
282         x = 0;
283         Serial.println();
284         Serial.println();
285
286         digitalWrite(LEDPIN, LOW);
287         attachInterrupt(digitalPinToInterrupt(pinIR), IR_Interrupt_Handler,
288 CHANGE);
289     }
290
291 #else
292     // variaveis para armazenar os valores dos sensores

```

```

292     int chuva = rainSensor();
293     int valorGas = analogRead(gas);
294     sensorExt.requestTemperatures();
295     sensorInt.requestTemperatures();
296     double tempExt = sensorExt.getTempCByIndex(0);
297     double tempInt = sensorInt.getTempCByIndex(0);
298
299     // se houver comunicacao com o modulo ethernet retorne a pagina WEB
300     word pos = ether.packetLoop(ether.packetReceive());
301     if(pos){
302         for(int c=pos; Ethernet::buffer[c]; c++)
303             Serial.print((char)Ethernet::buffer[c]);
304         Serial.println();
305         word n;
306         n = homePage(chuva, valorGas, tempExt, tempInt, alarm, count);
307         ether.httpServerReply(n);
308     }
309
310
311     tempo_atual = millis(); // atualiza o tempo atual
312
313     ativaAlarme(digitalRead(button)); // verifica se o botao foi pressionado
314
315     unsigned long tempoDecorrido = (tempo_atual - tempo_inicial); // calcula
316     o tempo decorrido
317
318     // atualiza a tela de acordo com o tempo decorrido
319     if(tempoDecorrido < tempo_tela1){
320         atualizaTela1();
321     }else if(tempoDecorrido < (tempo_tela1 + tempo_tela2)){
322         atualizaTela2(chuva, valorGas);
323     }else if(tempoDecorrido < (tempo_tela1 + tempo_tela2 + tempo_tela3)){
324         atualizaTela3();
325     }
326     else{
327         tempo_inicial = millis();
328     }
329 #endif
330 }
331 }
```

Listing 1: Código Fonte Arduino

5.4 Bibliotecas Usadas

Nesse projeto, utilizamos uma combinação de bibliotecas para alcançar nossos objetivos com mais agilidade. A Wire foi utilizada para a comunicação I2C com o LCD. A LiquidCrystal_I2C nós utilizamos para exibir informações no display LCD. A DallasTemperature é usada para receber os dados do sensor e para realizar a conversão do valor lido para a temperatura em graus Celsius.

A IRremote permite que enviamos sinais infrevermelhos, então a partir de um código RAW captados pelo receptor criamos um sinal de envio. A EtherCard foi útil para realizara a configuração do módulo ENC28J60, conectando o Arduino na rede Ethernet. Com ela foi possível utilizar protocolos de rede para exibir informações captadas pelos sensores.

Biblioteca	Versão
IRremote	2.2.1
DallasTemperature	3.11.0
LiquidCrystal_I2C	1.1.4
OneWire	2.3.7
EtherCard	1.1.0

6 Teste de Campo do protótipo do Smart Home 2560

6.1 Teste e Controle das variáveis ambientais pelo Smart Home 2560

Para realizar os testes, configuramos o sistema em um apartamento de $45m^2$, especificamente na sala, devido ao seu tamanho reduzido e proximidade com o ar-condicionado e a cozinha. Primeiramente, para testar o sensor de gás, aproximamos um isqueiro do sensor, o que resultou no disparo do alarme. Em seguida, para verificar o sensor de chuva, utilizamos um borrifador de água próximo a ele, e o LCD exibiu corretamente a informação de que estava chovendo.

A fim de avaliar a segurança do sistema, ativamos o alarme e passamos a uma distância considerável do sensor, simulando uma invasão. Nesse caso, o alarme foi disparado com sucesso, comprovando sua eficácia. Por fim, para testar o sensor de temperatura interna, aquecemos o sensor manualmente que ultrapassasse $32^\circ C$. Assim que a temperatura atingiu esse valor, o sensor enviou um sinal infravermelho, que ligou o ar-condicionado conforme programado.

6.2 Aferição e Calibração dos sensores

No código firmware, implementamos um intervalo de 5 segundos na função de configuração (setup) para permitir que os sensores sejam calibrados antes de começarem a medir. Além disso, ajustamos manualmente cada sensor usando uma chave Phillips para garantir que suas leituras estivessem alinhadas com as expectativas do projeto. Dessa forma, asseguramos que os sensores estivessem devidamente configurados e prontos para fornecer medições precisas.

6.3 Sistema de alimentação de energia

Para alimentar nosso sistema, optamos por utilizar uma fonte bivolt chaveada de 9V de tensão. Essa escolha foi feita levando em consideração a necessidade de fornecer energia estável para o funcionamento de todos os componentes do sistema. A fonte bivolt permite que seja alimentado tanto por uma rede elétrica de 110V quanto por uma de 220V, proporcionando uma maior flexibilidade.

7 Referências Bibliográficas

Referências

- [1] Arduino-IRremote. *Arduino-IRremote*. Disponível em: <<https://github.com/Arduino-IRremote/Arduino-IRremote>>. Acesso em: 05 de junho de 2023.
- [2] Miles Burton. *Dallas Temperature*. Disponível em: <<https://github.com/milesburton/Arduino-Temperature-Control-Library>>. Acesso em: 05 de junho de 2023.
- [3] Frank Brabander. *LiquidCrystal I2C*. Disponível em: <<https://github.com/marcoschwartz/LiquidCrystal-I2C>>. Acesso em: 05 de junho de 2023.
- [4] Nicholas Humfrey. *EtherCard*. Disponível em: <<https://github.com/njh/EtherCard>>. Acesso em: 05 de junho de 2023.
- [5] Fábio Souza. *Arduino MEGA 2560*. Disponível em: <<https://embarcados.com.br/arduino-mega-2560/>>. Acesso em: 05 de junho de 2023.

8 Apêndice

Para leitura e conhecimento que os sistemas são concebidos e desenvolvidos adotando uma metodologia que deve conter um ciclo de vida de sistema.

Na disciplina de Sistemas Embarcados, ofertada desde o início do cursos de graduação em Engenharia de Computação e Engenharia de Controle e Automação está sendo proposto um modelo de ciclo de vida para Projeto de Sistemas Embarcados - PSE. Devido à inexistência de um ciclo de vida para PSE na literatura da área esta proposta de modelo está baseada em literatura sobre desenvolvimento de projetos de softwares e prática de execução de projeto de pesquisa que envolvem o desenvolvimento de Sistemas Embarcados.

Portanto, uma abordagem por definição de etapas de atividades para o ciclo de vida com uma visão top-down, ou seja das atividades mais gerais para as atividades específicas serão apresentadas em fluxo a seguir.

Ciclo de vida de Projeto de Sistema Embarcado - PSE

Figura 5: Principais etapas da definição de um Sistema Embarcado

