CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Universidade Federal
de Campina Grande

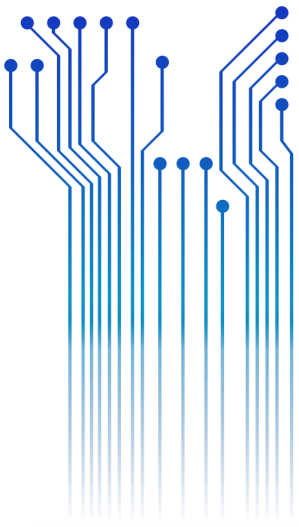
RODOLFO MEDEIROS ARAÚJO DA SILVA

Centro de Engenharia
Elétrica e Informática

TRABALHO DE CONCLUSÃO DE CURSO

SISTEMA DE MONITORAMENTO DE TEMPERATURA E UMIDADE PARA UMA CÂMARA ANECOICA

Departamento de
Engenharia Elétrica



Campina Grande
2015

RODOLFO MEDEIROS ARAÚJO DA SILVA

SISTEMA DE MONITORAMENTO DE TEMPERATURA E UMIDADE PARA UMA CÂMARA ANECOICA

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Instrumentação Eletrônica

Orientador:
Professor Tarso Vilela Ferreira, D. Sc.

Campina Grande
2015

RODOLFO MEDEIROS ARAÚJO DA SILVA

SISTEMA DE MONITORAMENTO DE TEMPERATURA E UMIDADE PARA UMA CÂMARA ANECOICA

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Instrumentação Eletrônica

Aprovado em / /

Professor Avaliador
Universidade Federal de Campina Grande
Avaliador

Professor Tarso Vilela Ferreira, D. Sc.
Universidade Federal de Campina Grande
Orientador, UFCG

Dedico este trabalho à minha mãe, pai, irmãos e familiares, assim como aos meus amigos que sempre estiveram ao meu lado, ao longo dessa jornada.

AGRADECIMENTOS

Agradeço, em primeiro lugar, a minha mãe Maria Simone por não ter medido esforços para garantir a minha educação.

Agradeço ao meu pai, Gilson Carlos, por ter sido importante durante toda a minha graduação.

Agradeço aos meus irmãos, Ana Luzia, Arthur e Heitor por estarem ao meu lado nos diversos momentos que passei durante essa longa jornada.

Agradeço aos meus tios, tias, avô e avós por serem importantes na minha formação tanto acadêmica, quanto pessoal.

Agradeço a Cecília Alves por ter estado ao meu lado desde os desenhos de Expressão Gráfica até os últimos momentos desse curso.

Agradeço aos meus amigos Priscilla, Hugo, Herbet, Rajiv, Kal-El, Bruno, Rodolpho, Martins, Anninha, Cândido, Léo, Guga, Fred e aos demais amigos, que fiz ao longo desse curso, por sempre estarem ao meu lado.

Agradeço aos meus amigos do 3º F, que me ajudaram a dar o primeiro passo na escolha da minha carreira, fazendo valer o apelido que tanto nos representou.

Agradeço aos meus amigos Laís, Marco e Vini por terem sido os melhores companheiros de piso.

Agradeço ao professor Dr. Tarso Vilela por ter me orientado durante o desenvolvimento das atividades desse trabalho.

Agradeço aos professores e funcionários do DEE por terem estado presente ao longo de toda essa jornada.

“ Se cheguei até aqui
foi porque me apoiei
em ombros de gigantes.”

Sir Isaac Newton.

RESUMO

Uma câmara anecoica é um ambiente blindado por paredes condutoras aterradas, revestido por absorvedores de rádio-frequência e isolado de fontes externas de ruído. Com isso, é possível simular as condições de um espaço de dimensão infinita, fazendo com que os experimentos realizados em seu interior não sejam comprometidos por influências exteriores. Este Trabalho de Conclusão de Curso apresenta um sistema de monitoramento de temperatura e umidade que utiliza a placa Arduino, uma plataforma de desenvolvimento *open source* de fácil programação, para medir e registrar as temperaturas e umidades internas e externas à câmara anecoica. Além disso, por meio de uma atuação direta com os aparelhos de ar condicionado do local monitorado, o sistema mantém iguais as temperaturas das áreas externa e interna da câmara, de forma automática e respeitando uma faixa pré-determinada de valores.

Palavras-chave: Câmara Anecoica, Arduino, Automatização.

LISTA DE ILUSTRAÇÕES

Figura 1 – Interior da câmara anecoica do LabMet-UFCG. 12
Figura 2 - Modelo Arduino Mega 2560. 14
Figura 3 - Diagrama funcional Arduino Mega 2560. 16
Figura 4 - Plataforma de *software* (IDE) do Arduino. 17
Figura 5 - Representação do sensor DHT11. 18
Figura 6 - Esquema de conexão do DHT11 18
Figura 7 - Conexão entre o Arduino e um display LCD. 19
Figura 8 - Representação do *Shield Ethernet.* 20
Figura 9 - Pinos utilizados pelo *Shield* Ethernet. 22
Figura 10 - Representação do módulo RTC DS1302. 23
Figura 11 - Diagrama do circuito do DS1302. 23
Figura 12 – Esquema de montagem para decodificação do sinal emitido pelo controle remoto. 25
Figura 13 – Esquema de montagem para emissão do sinal infravermelho. 25
Figura 14 – Esquema de montagem do protótipo desenvolvido. 26
Figura 15 – Protótipo desenvolvido. 27
Figura 16 - Resultados das medições de temperatura interna e externa sem atuação do controlador. 28
Figura 17 - Resultados das medições da umidade interna e externa sem atuação do controlador. 28
Figura 18 - Resultados das medições de temperatura interna e externa com atuação do controlador. 29
Figura 19 - Resultados das medições da umidade interna e externa com atuação do controlador. 29

LISTA DE ABREVIATURAS E SIGLAS

A/D	Analógico/Digital
CE	*Chip Enable*
DHT	*Detector of Humidity and Temperature*
E	*Enable*
IDE	*Integrated Development Enviroment*
I/O	*In/Out*
LCD	*Liquid Crystal Display*
MISO	*Master In Slave Out*
MOSI	*Master Out Slave In*
R/W	*Read/Write*
RS	*Register Selector*
RST	*Reset*
RTC	*Real Time Clock*
SCLK	*Serial Clock*
SPI	*Serial Peripheral Interface*
SS	*Slave Select*

SUMÁRIO

1 Introdução 12
1.1 Objetivos 13
1.2 Estrutura do Trabalho 13
2 Embasamento Teórico 14
2.1 Arduino 14
2.1.1 Características de *Hardware* 15
2.1.2 Características de *Software* 16
2.2 Sensor de Temperatura e Umidade – DHT11 17
2.3 LCD 20x4 19
2.4 Arduino Ethernet Shield 20
2.5 Módulo Relógio de Tempo Real DS1302 22
3 Desenvolvimento do Protótipo 24
4 Resultados 27
5 Conclusão 30
Referências 31
APÊNDICE A – Código para decodificação do sinal IR 32
APÊNDICE B – Código final para controle de umidade e temperatura 34

1 INTRODUÇÃO

Uma câmara anecoica é um ambiente blindado por paredes condutoras aterradas, revestido por absorvedores de radiofrequências e isolado de fontes externas de ruído. A combinação desses aspectos significa que esse ambiente simula as condições de um espaço de dimensão infinita, o que é bastante útil quando influências exteriores comprometem os resultados de experimentos, como ensaios de interferência e susceptibilidade eletromagnéticas realizados em campo aberto.

Para garantir a não ocorrência de reflexões das ondas eletromagnéticas, o interior da câmara anecoica é totalmente revestido por estruturas conhecidas como absorvedores. Essas estruturas são constituídas por uma espuma emborrachada, impregnada com misturas controladas de carbono e ferro, não podendo ser nem um bom condutor nem um bom isolante, já que ambos não absorvem qualquer tipo de potência.

Para uma maior eficiência da câmara, eles devem ter formato piramidal, como visto na Figura 1, com suas pontas direcionadas para dentro do ambiente, maximizando o número de reflexões da onda dentro da estrutura e minimizando sua intensidade. Esses absorvedores, contudo, requerem níveis de temperatura e umidade dentro de certos limites, a fim de que sua vida útil seja prolongada. Além disso, é desejável que a temperatura e a umidade internas e externas a câmara sejam similares [1].

Figura 1 – Interior da câmara anecoica do LabMet-UFCG.

Fonte: O próprio autor, 2015.

Assim, propõe-se neste trabalho desenvolver um sistema de medição de temperatura e umidade baseado no microcontrolador ATmega2560. Além de medir e registrar as temperaturas e umidades internas e externas à câmara, o sistema deverá atuar junto aos aparelhos de ar condicionado do laboratório, a fim de manter a temperatura e a umidade nos níveis recomendados.

1.1 OBJETIVOS

O objetivo principal deste trabalho é desenvolver um protótipo do sistema de medição de temperatura e umidade de uma câmara anecoica, capaz ainda de atuar no controle do ar-condicionado responsável pela refrigeração da mesma, garantindo assim que temperatura e umidade conservem-se dentro de faixas determinadas. O sistema deve ainda tornar o acesso aos dados de temperatura e umidade acessíveis aos usuários.

1.2 ESTRUTURA DO TRABALHO

Este trabalho de conclusão de curso está dividido em quatro capítulos. O primeiro capítulo destina-se à parte introdutória. No segundo capítulo é feito o embasamento teórico do trabalho, onde serão abordados os temas essenciais para o entendimento do tema proposto. O terceiro capítulo destina-se a explicar as etapas do processo de desenvolvimento do protótipo. No quarto capítulo, os testes realizados e os resultados obtidos são mostrados. No quinto capítulo as conclusões e considerações finais são apresentadas.

2 EMBASAMENTO TEÓRICO

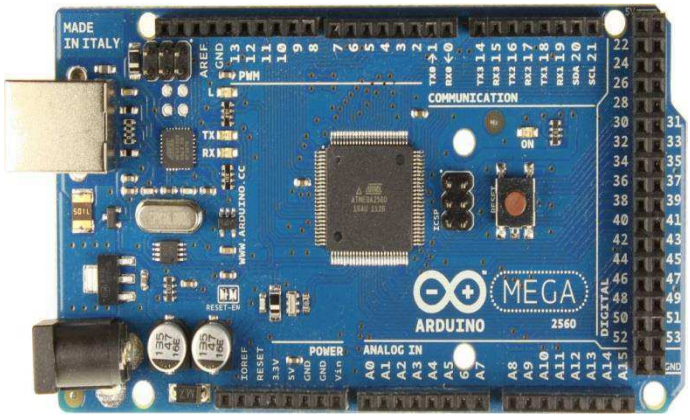
Para o entendimento do método utilizado no desenvolvimento do protótipo que realizará o controle de temperatura e umidade da câmara anecoica, é necessário tecer alguns comentários sobre o Arduino, o sensor DHT11, o LCD 20x4, o RTC e o módulo *Shield Ethernet*.

2.1 ARDUINO

A plataforma Arduino foi desenvolvida em 2005 na Itália, pela equipe liderada pelo professor Massimo Banzi e constituída também por David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis. Seu objetivo era abstrair ao máximo a parte eletrônica do desenvolvimento de projetos, mensurando o mundo físico através do processamento de entradas e saídas, facilitando assim o desenvolvimento de projetos por entusiastas. Seu nome tem origem na palavra germânica *hardwin*, composta pelas palavras *hard* (forte) e *win* (amigo), e adaptado para o italiano, transformou-se em Arduino [2].

Atualmente já existem diversas versões de placas Arduino, sendo uma delas a Arduino MEGA, que é utilizada neste projeto e exposta na Figura 2.

Figura 2 - Modelo Arduino Mega 2560.



Fonte: <https://www.arduino.cc/en/Main/ArduinoBoardMega2560>, 2015.

Uma das principais características desta plataforma é sua documentação aberta, com todos os esquemáticos de *hardware* e diversas bibliotecas disponibilizadas

livremente, possibilitando que qualquer pessoa replique a placa e crie a sua própria versão. Outro ponto destacado é seu caráter modular, caracterizado pelos *shields*, placas com circuitos específicos que podem ser acoplados ao Arduino, expandindo suas funcionalidades [3].

2.1.1 CARACTERÍSTICAS DE *HARDWARE*

Podemos ainda destacar como principais características do Arduino MEGA:

- Microcontrolador ATmega2560;
- 54 entradas/saídas digitais (das quais 16 podem ser utilizadas como saída PWM);
- 16 entradas analógicas com 10 bits de resolução cada;
- Tensão de operação: 5 V;
- Conexão USB e ICSP;
- Comunicação serial, SPI e i2c;
- 16 MHz de *clock*.

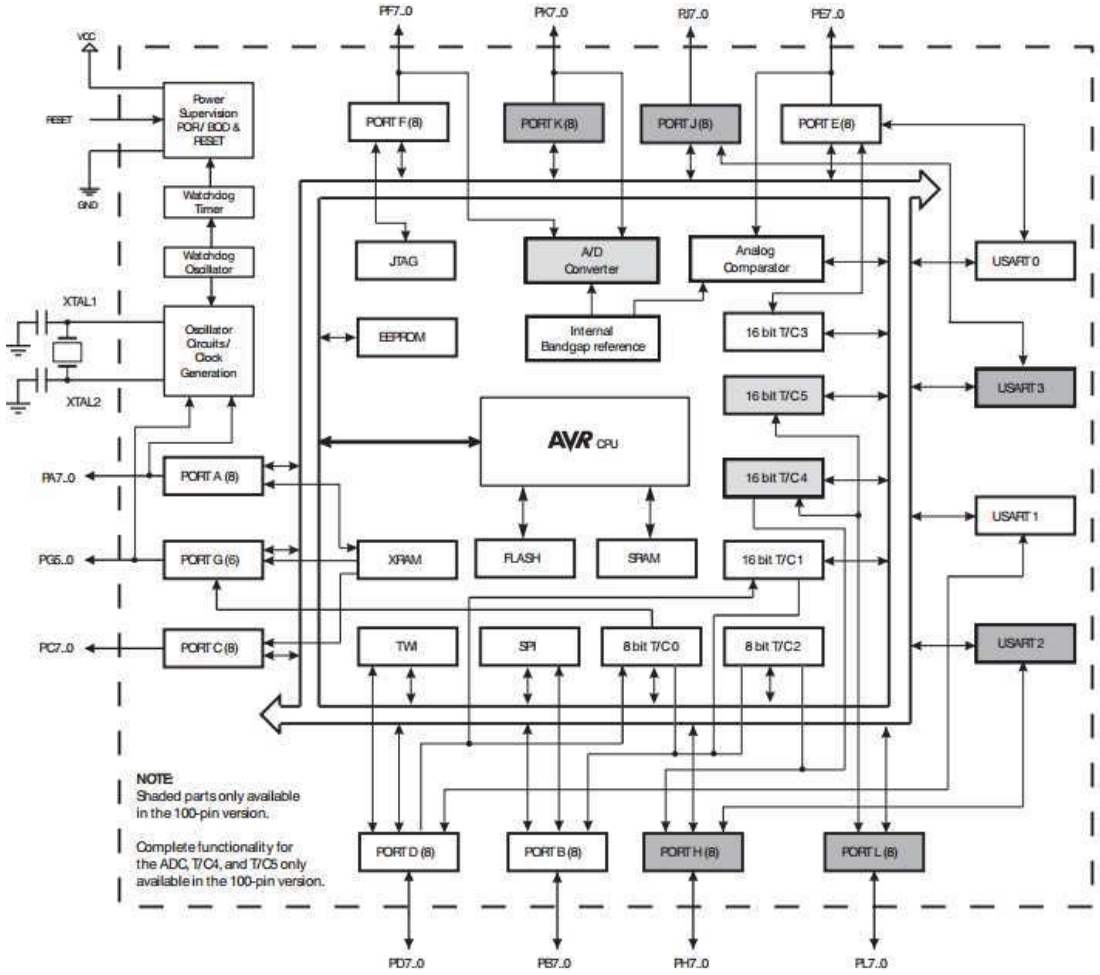
O microcontrolador presente como unidade central de processamento da placa Arduino, o ATmega2560, pode ter seu funcionamento resumido através do diagrama funcional representado na Figura 3, que apresenta as estruturas de *hardware* do dispositivo [4].

O ATmega2560 possui 32 registradores de uso geral, três temporizadores/contadores com modos de comparação, interrupções interna e externa serial programável, uma porta SPI, *watchdog timer* com oscilador interno, e conversor A/D de 10 bits, representando níveis de tensão distintos.

Ele é extremamente versátil, com elevada capacidade de processamento e conjunto diversificado de periféricos. Quando utilizado na plataforma Arduino, se torna a solução ideal para o projeto proposto nesse trabalho, haja vista que segundo [5]

> "*É portátil, tem quantidade suficiente de entradas e saídas, documentação aberta e disponível sem custos adicionais, além do seu caráter modular que permite facilmente alterações na proposta do projeto e expansão de funcionalidades*".

Figura 3 - Diagrama funcional Arduino Mega 2560.



Fonte: <https://www.arduino.cc/en/Main/ArduinoBoardMega2560>, 2015.

2.1.2 CARACTERÍSTICAS DE *SOFTWARE*

A plataforma de *software* do Arduino, modelada na linguagem de programação *Processing*, denominada IDE, trata-se de um programa executado em um computador, que permite a codificação de programas em linguagem específica, denominados *sketches*, a serem carregados na plataforma do Arduino [6].

É importante destacar que o microcontrolador ATmega2560 trabalha com programação em C. A IDE do Arduino traduz o *sketch* escrito pelo desenvolvedor em linguagem C e, na sequência, realiza a gravação do código no microcontrolador.

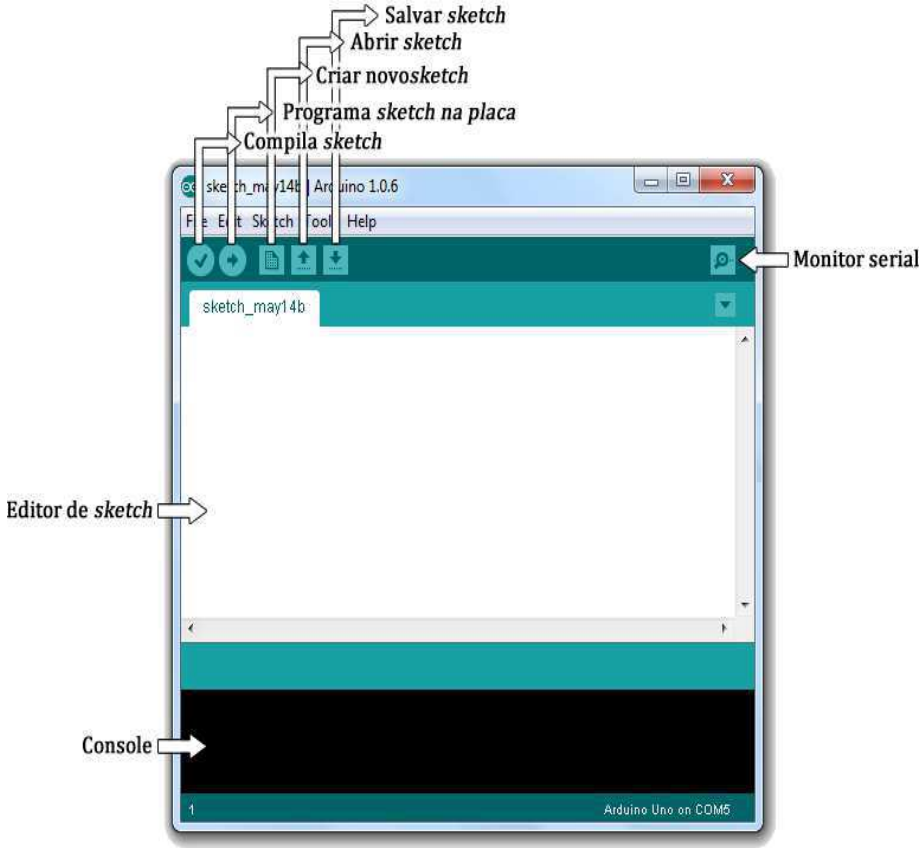
Desta maneira, o ciclo de programação de um Arduino pode ser resumido nesses passos:

1. Escrever o *sketch*;

2. Compilar o *sketch*;
3. Ligar a placa a uma porta USB do computador;
4. Enviar o *sketch* à placa, através da porta USB;
5. A placa executa o programa.

A IDE do Arduino possui *interface* amigável e familiar, podendo ser executada em qualquer sistema operacional. Sua *interface* está exposta na Figura 4.

Figura 4 - Plataforma de *software* (IDE) do Arduino.

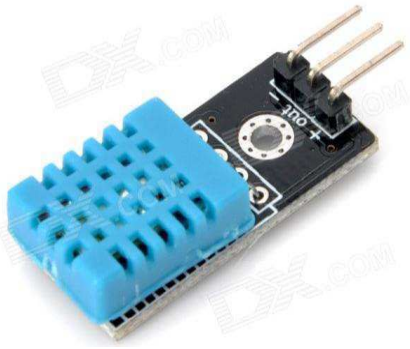


Fonte: O próprio autor, 2015.

2.2 SENSOR DE TEMPERATURA E UMIDADE – DHT11

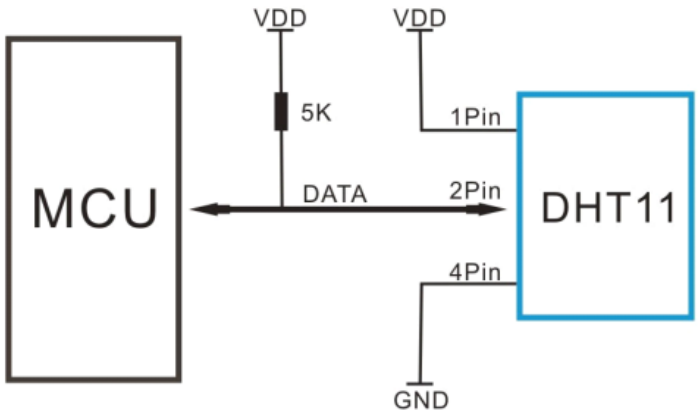
O sensor DHT11 é um sensor de temperatura e umidade que permite realizar leituras de temperaturas entre 0 e 50 °C, e umidade relativa entre 20 e 90%, apresentando ainda precisão de ± 2 °C e ± 5%, o que satisfaz as necessidades do projeto [7]. Este dispositivo opera com tensão de 3,3 a 5 V DC e, conforme indicado no seu esquema elétrico (Figura 6), utiliza o protocolo de comunicação 1-*wire* [8].

Figura 5 - Representação do sensor DHT11.



Fonte: <http://www.arduinoecia.com.br/2013/05/sensor-de-umidade-e-temperatura-dht11.html>, 2015.

Figura 6 - Esquema de conexão do DHT11.



Fonte: <http://rduinostar.com/documentacion/datasheets/dht11-overview/>, 2015.

Um único processo de comunicação do DHT11 dura aproximadamente 4 ms. Os dados consistem de 40 bits, sendo 16 bits para o valor da umidade, 16 bits para o da temperatura e 8 bits de *checksum*. Tendo posse destes dados fornecidos pelo fabricante do sensor, é possível enviar o sinal de início para o sensor, que responderá confirmando o recebimento do comando, e em seguida irá enviar os dados da sua leitura, pelo mesmo canal pelo qual recebeu seu sinal de comando.

Para o Arduino, existe a biblioteca *dht*, de documentação aberta e disponibilizada gratuitamente, que abstrai a parte de comandos, pulsos de sinais enviados e leitura de dados recebidos, facilitando o trabalho de leitura do sensor [7].

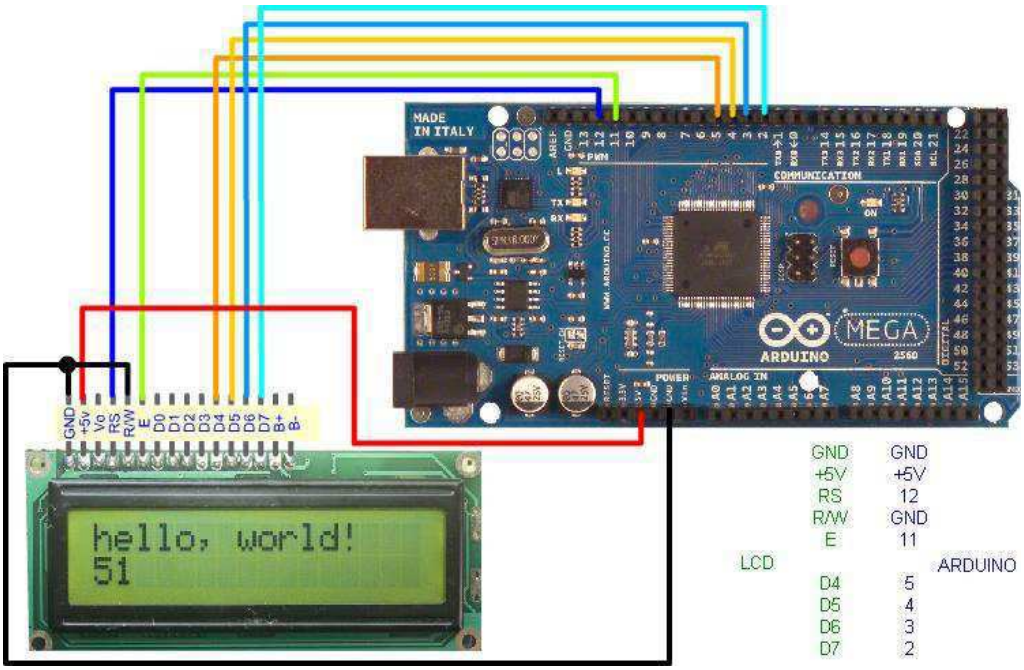
A utilização deste sensor possibilita ao projeto um monitoramento constante de variáveis importantes do ambiente no qual se insere a câmara anecoica.

2.3 LCD 20x4

O display de cristal líquido (LCD) será o responsável por permitir a visualização das leituras do sensor DHT11. Para tal, ele possui a tabela de caracteres ASCII na sua memória.

Este dispositivo opera com tensão de 5 V DC e foi interligado ao protótipo da forma representada na Figura 7.

Figura 7 - Conexão entre o Arduino e um display LCD.



Fonte: <http://www.arduinoecia.com.br/2014/06/arduino-display-lcd-20x4.html>, 2015.

Para o funcionamento correto, o display necessita receber três sinais do micro controlador, sendo eles:

- E: Este sinal permite acessar o display através dos sinais R/W e RS;
- R/W: Determina a direção que os dados fluem entre o LCD e o microcontrolador.
- RS: De acordo com os dados nesse controlador, o LCD interpreta o tipo de dado que está sendo enviado pelo microcontrolador.

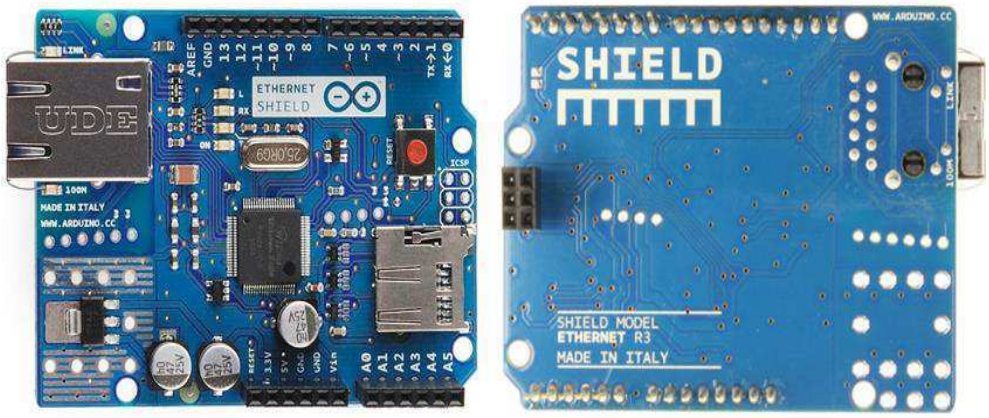
A comunidade de desenvolvedores para Arduino projetaram uma biblioteca de documentação aberta e gratuita, denominada LiquidCrystal, que facilita a comunicação entre a placa e o display [9].

2.4 ARDUINO ETHERNET SHIELD

Conforme supramencionado, uma das principais vantagens da plataforma Arduino é seu caráter modular, cujos principais expoentes são seus *shields*, módulos que, quando acoplados ao Arduino, expandem suas funcionalidades. São placas de circuito impresso que agregam recursos de *hardware* ao microcontrolador, e apresentam como principal característica o mesmo *design* da placa Arduino UNO, para um encaixe perfeito.

Para esse projeto, foi utilizado o Arduino *Ethernet Shield*, representado na Figura 8, e responsável por tornar o Arduino capaz de se conectar à internet. Como todas as placas Arduino, este *shield* possui toda a sua documentação aberta e disponível gratuitamente. Os principais motivos para a escolha deste *shield* para o projeto foram: a presença de um adaptador para cartão de memória SD, e a possibilidade de expansão de funcionalidades do projeto.

Figura 8 - Representação do *Shield Ethernet.*



Fonte: <https://www.arduino.cc/en/Main/ArduinoEthernetShield>, 2015.

Esta placa é baseada no chip *ethernet Wiznet* W5100, responsável por prover acesso à internet utilizando o protocolo de comunicação TCP/IP, e tem ainda como características que merecem destaque:

- Velocidade de conexão 10/100 Mb;
- Conector padrão RJ45;
- Conexão com o Arduino via protocolo SPI.

O SPI é um protocolo de dados serial síncrono, utilizado por microcontroladores para comunicação rápida com um ou mais dispositivos periféricos em curtas distâncias, além de poder ser usado para comunicação entre dois microcontroladores, e sendo usado pelo *shield* também para armazenar dados e realizar a leitura do cartão de memória SD presente na placa.

Em uma comunicação SPI, sempre há um mestre e um ou mais escravos, que se comunicam entre si através de uma conexão *full-duplex*. Os seguintes canais constituem a conexão SPI:

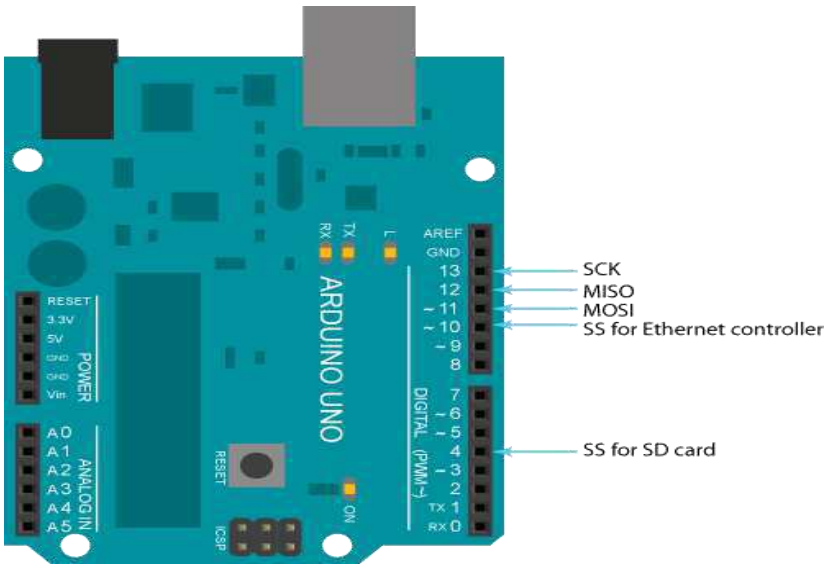
- MOSI: Canal responsável pelo envio de dados do mestre para o escravo;
- MISO: Canal responsável pelo envio de dados do escravo para o mestre;
- SCK: Canal para envio do sinal de *clock*;
- SS: Canal utilizado pelo mestre para habilitar ou desabilitar a conexão com o escravo. Se na conexão existir mais de um escravo, todos compartilham os canais citados anteriormente, com exceção deste. Cada escravo possui seu SS. Quando este pino estiver em nível baixo, há conexão com o mestre e quando estiver em nível alto, o escravo ignora as informações do barramento.

Na comunicação SPI são utilizados dois registradores de deslocamento para a troca de informações, um pertencente ao mestre e o outro ao escravo. Os bits são deslocados e enviados um por vez, entre os registradores. À medida que um bit é transmitido pelo canal MOSI, outro é recebido pelo canal MISO, e após todos serem enviados, os registradores do mestre e do escravo estarão com os dados trocados [10].

Para utilizar o *shield ethernet* com o Arduino MEGA 2560, faz-se necessário o uso de três bibliotecas de *software* disponibilizadas gratuitamente no site do Arduino, e que são instaladas junto com a IDE de programação dos *sketches*. Essas bibliotecas são a Ethernet, cujas funções não serão utilizadas a princípio no projeto; a SPI, responsável por ler e escrever comandos SPI; e a SD, que contém as funções necessárias para trabalhar com o cartão de memória SD. A Figura 9 apresenta quais pinos do Arduino o

shield utiliza, desta maneira estes pinos não podem ser utilizados para outra função no projeto, a fim de se evitar conflitos e mau funcionamento [11].

Figura 9 - Pinos utilizados pelo *Shield* Ethernet.



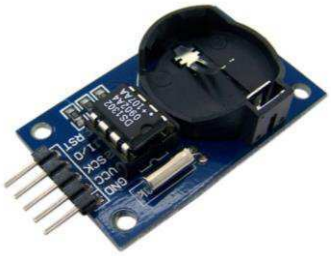
Fonte: <https://www.arduino.cc/en/Main/ArduinoEthernetShield>, 2015.

2.5 MÓDULO RELÓGIO DE TEMPO REAL DS1302

Como será realizado o monitoramento de sensores, aquisição e armazenamento de dados, é importante a utilização de um relógio em tempo real no sistema, a fim de se armazenar horários em que eventos importantes aconteceram, além de controlar o monitoramento dos sensores, haja vista que a leitura dos mesmos é feita a cada 30 segundos, contagem esta a ser feita pelo relógio.

Para este projeto, foi escolhido o módulo RTC DS1302, exibido na Figura 10, que consiste em um chip DS1302, com funções de hora, data e calendário. Ele é capaz de fornecer informações de data, mês e ano, e ajusta automaticamente dados referentes aos meses com menos de 31 dias, além de anos bissextos. Desta maneira, ele é mais que suficiente para atender as necessidades do projeto [12].

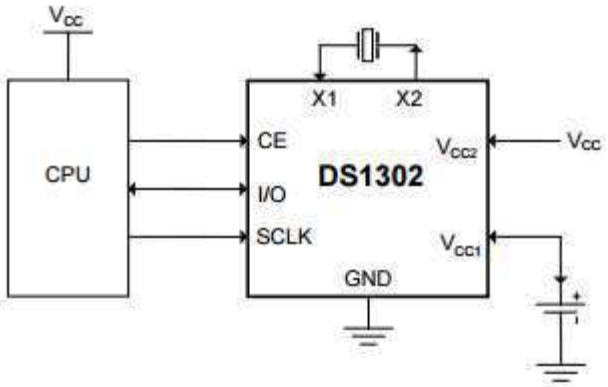
Figura 10 - Representação do módulo RTC DS1302.



Fonte: <http://playground.arduino.cc/Main/DS1302>, 2015.

O diagrama de um circuito típico do DS1302 é exposto na Figura 11. É importante destacar a presença de duas fontes de alimentação. Uma delas, a V_{cc2}, é a alimentação fornecida pelo Arduino, e a outra, V_{cc1}, é a tensão de 3,3 V fornecida pela bateria externa, responsável por manter o relógio funcionando mesmo quando o Arduino estiver desligado. Deve-se usar um oscilador cristal de 32.768 kHz. O DS1302 utiliza três pinos para realizar a comunicação com o microcontrolador: CE (ou RST), I/O (ou DATA) e SCLK .

Figura 11 - Diagrama do circuito do DS1302.



Fonte: <http://playground.arduino.cc/Main/DS1302>, 2015.

A comunidade de desenvolvedores para Arduino projetou uma biblioteca de documentação aberta e gratuita, denominada virtuabotixRTC, que facilita o trabalho de realizar a comunicação byte a byte com o DS1302, configurá-lo e extrair os dados [13].

Esse é um componente de baixo custo e, apesar de não ter uma vida útil longa e apresentar problemas em longo prazo, em curto prazo, como protótipo, satisfaz todas as necessidades do projeto.

3 DESENVOLVIMENTO DO PROTÓTIPO

Com o intuito de validar o modelo de sistema de monitoramento de temperatura e umidade proposto, foi projetado um protótipo contendo todos os elementos integrantes do projeto.

Teve-se como objetivo verificar o correto funcionamento do sistema em todas as suas etapas, desde a aquisição de dados, processamento de dados, saídas e acionamento dos atuadores.

O método utilizado para controlar o ar condicionado contido na câmara anecoica foi o de clonar os sinais emitidos pelo controle remoto.

Em um estudo preliminar, observou-se que para cada temperatura do ar condicionado, o controle emitia um sinal diferente. A primeira etapa do desenvolvimento do protótipo consistiu em decodificar cada sinal emitido pelo controle remoto. Isso foi possível utilizando o receptor infravermelho. O esquema montado para a realização dessa etapa está mostrado na Figura 12. O código elaborado para esta etapa encontra-se no Apêndice A.

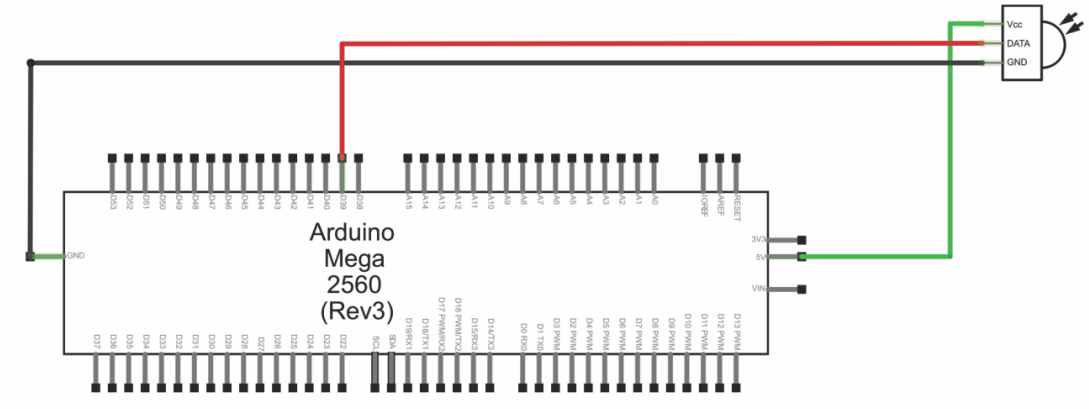
Com os dados do sinal decodificado, o protótipo começou a ser montado. Em um primeiro momento, os sensores DHT11, o *shield ethernet* e o RTC foram conectados ao Arduino, de forma que se tornou possível salvar em um arquivo txt os dados de temperatura e umidade captados, em seguida, o *display LCD* foi conectado, possibilitando a visualização das temperaturas e umidades internas e externas.

Por fim, conectou-se o emissor infravermelho (LED), que tem a função de emitir os sinais de controle de temperatura para o ar-condicionado e permitir a equivalência entre as temperaturas interna e externa à câmara. Esse emissor, entretanto, possui uma limitação quanto ao seu alcance. Após sua conexão ao circuito, foi possível observar que o aparelho de ar-condicionado só respondia ao sinal enviado pelo emissor quando o mesmo se encontrava a uma distância de até 30 cm.

Para superar essa limitação de distância, foi montado o circuito apresentado na Figura 13. A associação de resistores de 27 Ω e 10 kΩ e o transistor BC 547 com o emissor infravermelho, conforme apresentado, permitiu que o alcance para atuação do aparelho de ar-condicionado chegasse a 3 metros.

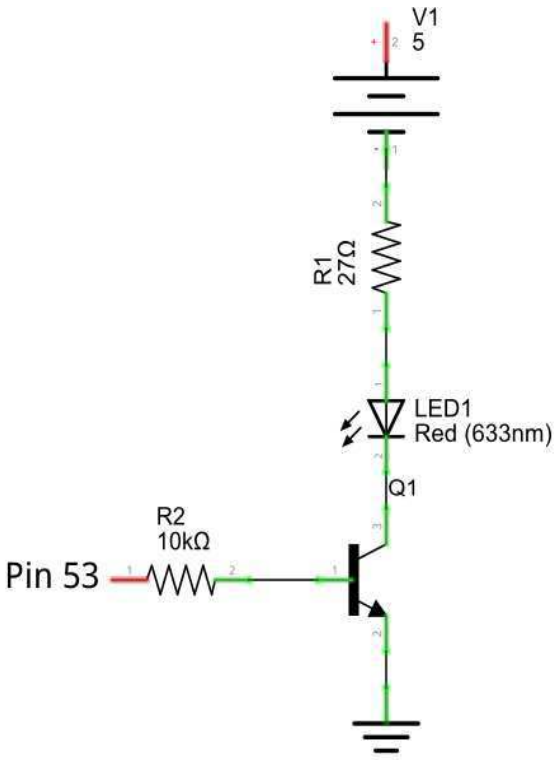
Uma vez montado, o protótipo se tornou capaz de emular os sinais decodificados do controle do ar-condicionado, e com isso foi possível realizar testes para saber se os resultados obtidos estariam de acordo com o projeto proposto inicialmente. Os resultados obtidos são apresentados adiante. O esquema do protótipo montado está mostrado na Figura 14. O código elaborado para o seu funcionamento encontra-se no Apêndice B.

Figura 12 – Esquema de montagem para decodificação do sinal emitido pelo controle remoto.



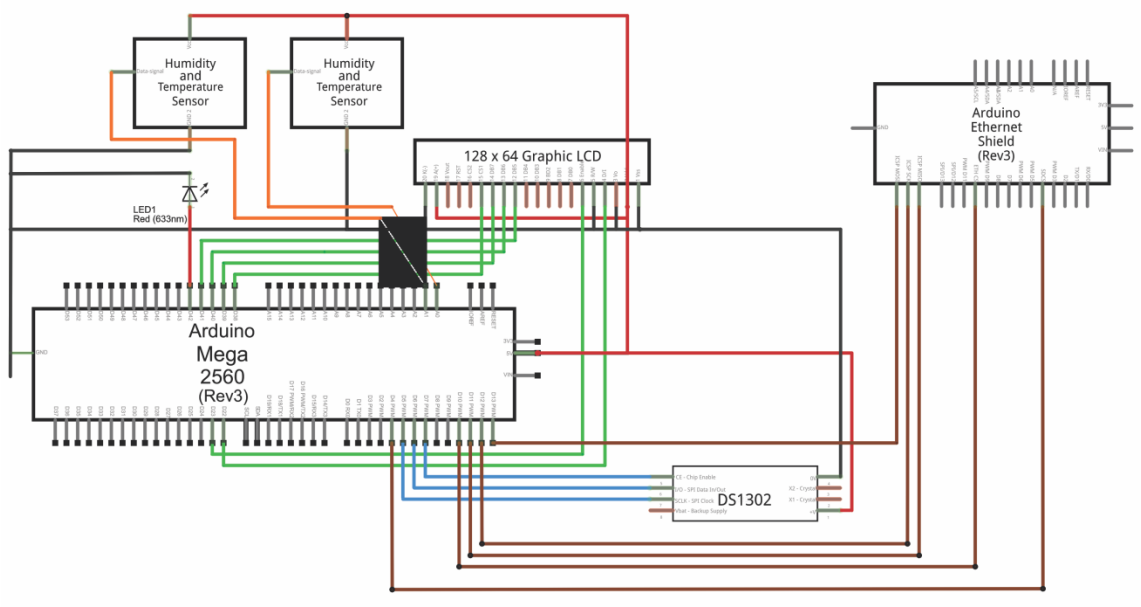
Fonte: O próprio autor, 2015.

Figura 13 – Esquema de montagem para emissão do sinal infravermelho.



Fonte: O próprio autor, 2015.

Figura 14 – Esquema de montagem do protótipo desenvolvido.



Fonte: O próprio autor, 2015.

4 RESULTADOS

Com o protótipo já montado (Figura 15), foram realizados testes para validação no laboratório onde a câmara anecoica está localizada.

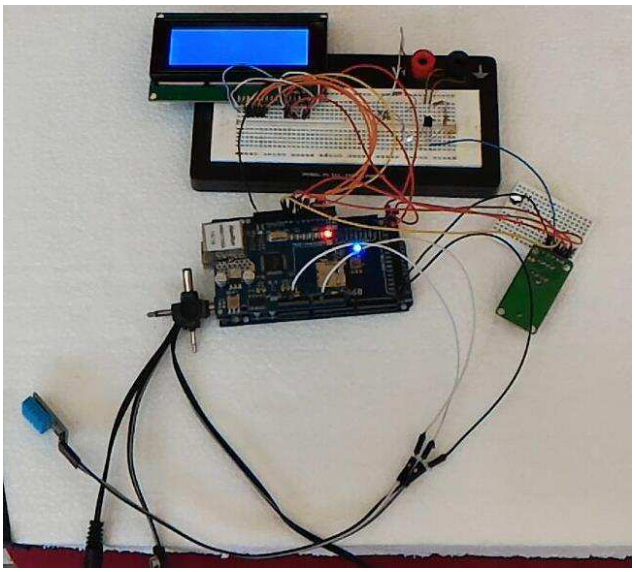
Para a obtenção dos dados referentes ao interior da câmara, o sensor foi posicionado próximo à porta. O ar condicionado que controla a temperatura interna da câmara não estava funcionando, logo as medições de temperatura e umidade se mantiveram constantes durante o período medido.

Em seguida, foram feitas as medições da temperatura e umidade ambiente do laboratório por um período de oito horas. Os valores obtidos foram salvos no SD *Card* e são apresentados nas Figuras 16 e 17.

Para a realização de um segundo teste, a temperatura externa foi aumentada manualmente. O protótipo foi configurado de forma que atuaria caso a temperatura do ambiente fosse diferente da temperatura interna da câmara. Assim, ele detectou a diferença de temperaturas e atuou, reduzindo a temperatura do ambiente externo à câmara. As medições realizadas são apresentadas nas Figuras 18 e 19.

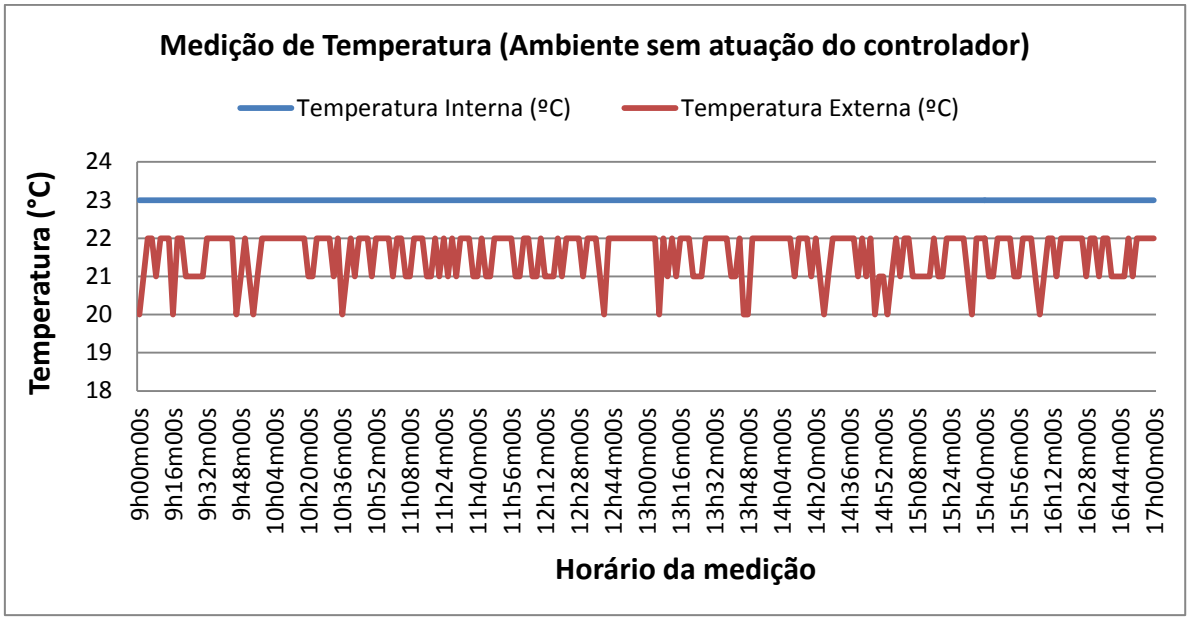
No caso da umidade, os resultados obtidos mostraram que o controle não foi tão preciso quanto o da temperatura, porém eles ficaram dentro da faixa de tolerância do sensor utilizado.

Figura 15 – Protótipo desenvolvido.



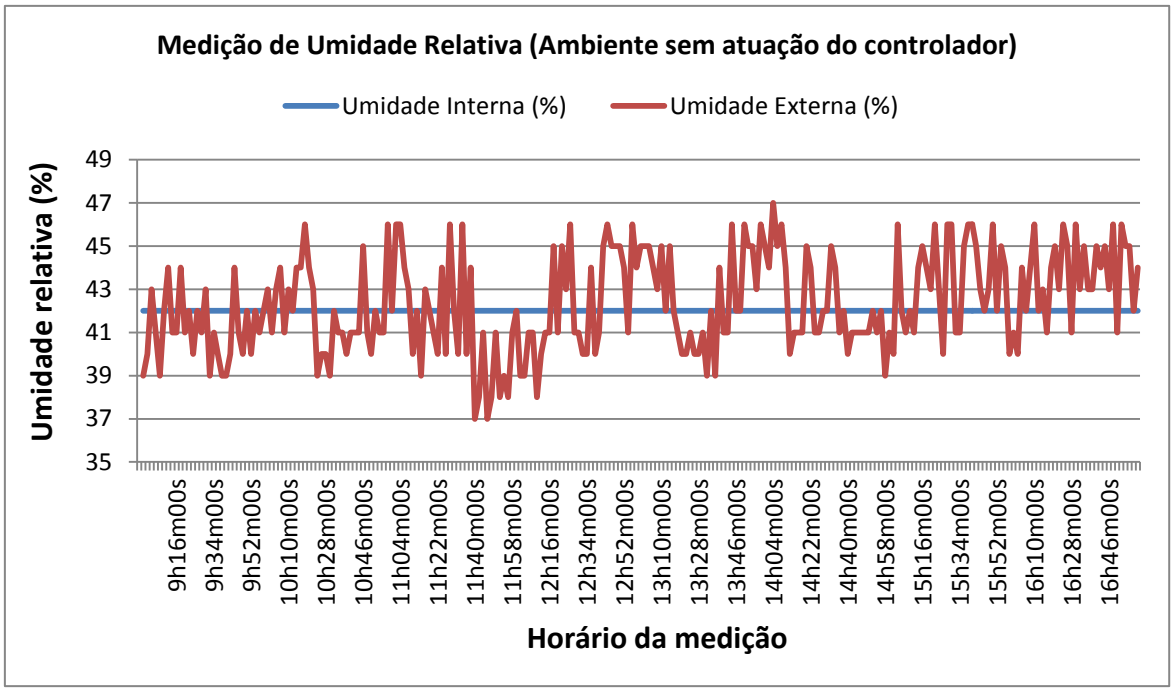
Fonte: O próprio autor, 2015.

Figura 16 - Resultados das medições de temperatura interna e externa sem atuação do controlador.



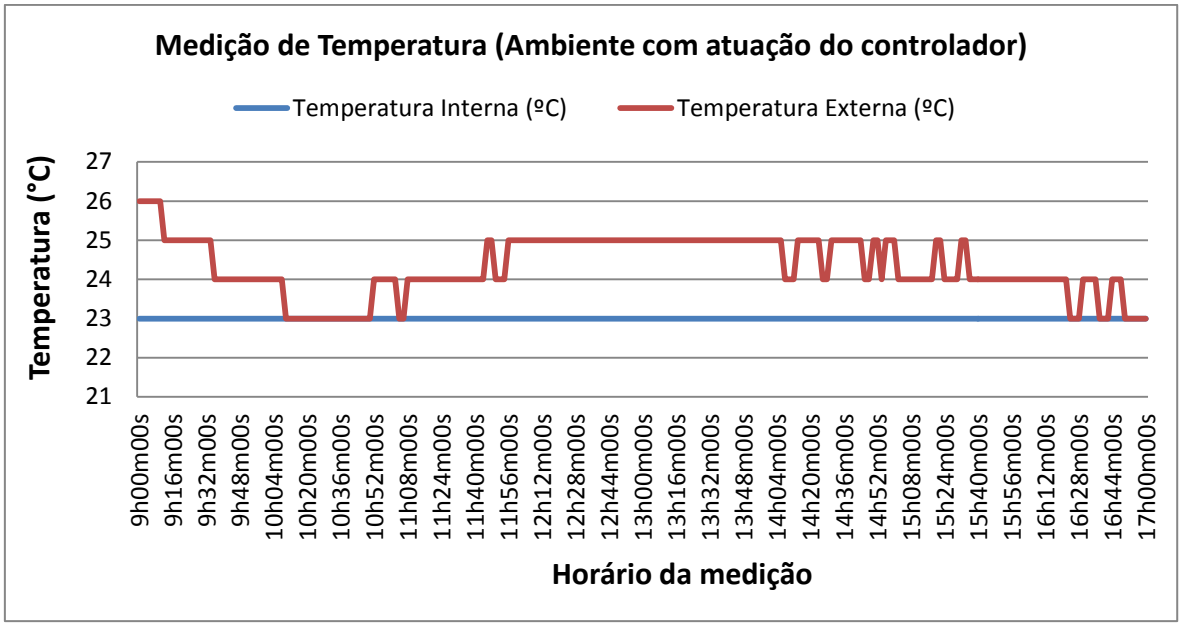
Fonte: O próprio autor, 2015.

Figura 17 - Resultados das medições da umidade interna e externa sem atuação do controlador.



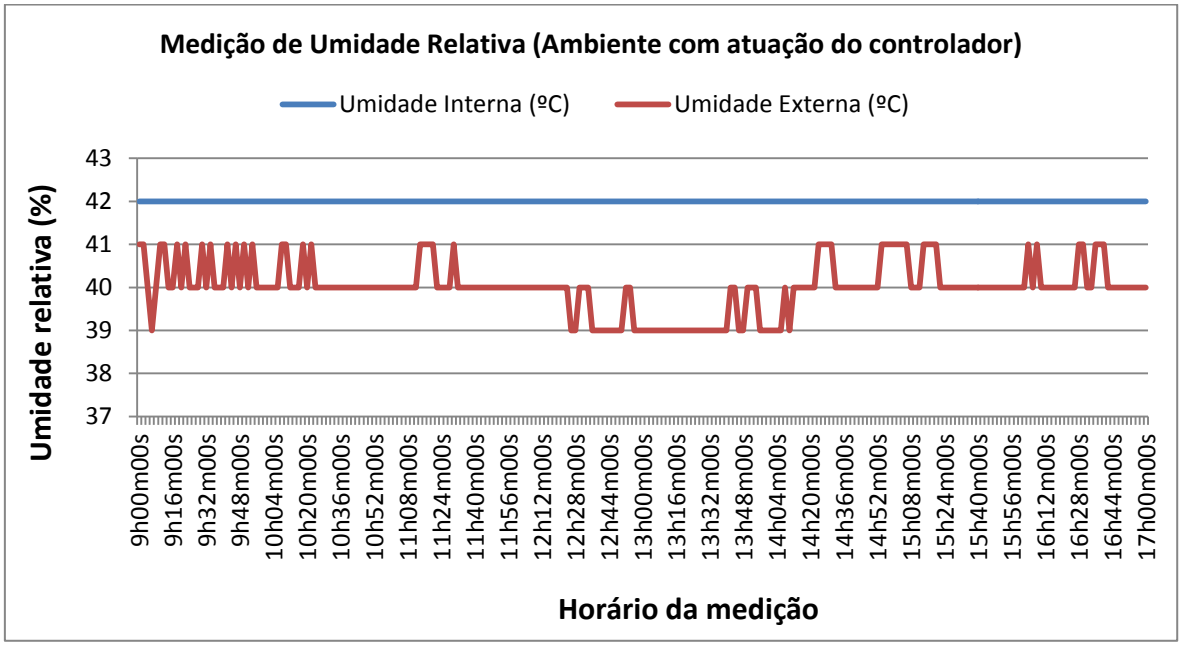
Fonte: O próprio autor, 2015.

Figura 18 - Resultados das medições de temperatura interna e externa com atuação do controlador.



Fonte: O próprio autor, 2015.

Figura 19 - Resultados das medições da umidade interna e externa com atuação do controlador.



Fonte: O próprio autor, 2015.

5 CONCLUSÃO

Neste trabalho foi realizado um projeto de um sistema de monitoramento de temperatura e umidade de uma câmara anecoica utilizando um Arduino como unidade central de processamento, além de periféricos como sensores e atuadores.

Foi desenvolvido um protótipo para proporcionar a validação do sistema projetado, de forma que os resultados obtidos nos testes foram bastante satisfatórios, haja vista que ficaram numa zona de tolerância dos sensores utilizados.

O presente trabalho aborda diversos aspectos de um projeto de engenharia elétrica, ao se debater sobre temas como eletrônica e microeletrônica, arquiteturas de sistemas digitais e programação de processadores. Também contempla a montagem experimental, resultando no desenvolvimento de um protótipo que pode ser usado como modelo para o desenvolvimento de outros produtos e no levantamento para construção de protótipos e atividades experimentais, todos esses aspectos importantes no desenvolvimento e formação de um profissional de engenharia elétrica.

Para projetos futuros, existem alguns pontos no sistema que poderão ser melhorados. São eles:

- Melhor aproveitamento do *shield ethernet*, através da criação de uma *interface* via *web* com monitoramento de dados e controle do sistema;
- Criação de um aplicativo para Android, melhorando a interface do sistema com o *smartphone*;
- Ampliação para um sistema com mais sensores e atuadores, para melhorar a medição dos dados e atuação no sistema.

REFERÊNCIAS

[1] WEINZIERL, D., Estudo de câmaras de teste de Compatibilidade Eletromagnética (CEM) através do método de Modelagem por Linhas de Transmissão (TLM). Tese (Doutorado em Engenharia Elétrica). Universidade Federal de Santa Catarina, Florianópolis, 2004. 133 p..

[2] KUSHNER, David. **The Making of Arduino**. IEEE Spectrum. Disponível em: http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino. Acesso em 28/11/2015.

[3] **Arduino Mega**. Disponível em: http://www.arduino.cc/en/main/arduinoBoardMega. Acesso em 28/11/2015.

[4] McROBERTS, M. **Arduino básico**. Trad. Rafael Zanolli. São Paulo: Novatec, 2011.

[5] ATMEL Corporation. **ATmega48A/48PA/88A/88PA/168A/168PA/328/328P Data Sheet**. [S.l.] 2010.

[6] **A short introduction to the Processing software and projects from the community**. Disponível em: https://processing.org/overview/. Acesso em 28/11/2015.

[7] **Moisture Sensor (SKU:SEN0114)**. Disponível em: http://www.dfrobot.com/wiki/. Acesso em 28/11/2015.

[8] D-Robotics UK. **DHT11 Umidity & Temperature Sensor Data Sheet**. [S.l.] 2010.

[9] LCD 20x4. Disponível em: http://centralavr.blogspot.com.br/2011/04/utilizando-um-display-de-lcd-com-o.html. Acesso em 28/11/2015.

[10] **Arduino Ethernet Shield**. Disponível em: http://www.arduino.cc/en/Main/ArduinoEthernetShield. Acesso em 29/11/2015.

[11] **Ethernet Library**. Disponível em: http://www.arduino.cc/en/reference/ethernet. Acesso em 29/11/2015.

[12] Maxim Integrated/Dallas Semicondutor. **DS1302 Trickle-Charge Timekeeping Chip**. [S.l.] 2015.

[13] **DS1302 RTC**. Disponível em: http://playground.arduino.cc/Main/DS1302. Acesso em 29/11/2015.

APÊNDICE A – CÓDIGO PARA DECODIFICAÇÃO DO SINAL IR

```
#define IRpin_PIN  PINA
#define IRpin   0
// the maximum pulse we'll listen for - 65 milliseconds is
a long time
#define MAXPULSE 65000
// what our timing resolution should be, larger is better
// as its more 'precise' - but too large and you wont get
// accurate timing
#define RESOLUTION 20
// we will store up to 100 pulse pairs (this is -a lot-)
uint16_t pulses[100][2]; // pair is high and low pulse
uint8_t currentpulse = 0; // index for pulses we're storing
void setup(void) {
Serial.begin(9600);
Serial.println("Ready to decode IR!");
}
void loop(void) {
uint16_t highpulse, lowpulse; // temporary storage timing
highpulse = lowpulse = 0; // start out with no pulse length
// while (digitalRead(IRpin)) { // this is too slow!
while(IRpin_PIN & _BV(IRpin)) {
// pin is still HIGH
// count off another few microseconds
highpulse++;
delayMicroseconds(RESOLUTION);
// If the pulse is too long, we 'timed out' - either
nothing
// was received or the code is finished, so print what
// we've grabbed so far, and then reset
if((highpulse >= MAXPULSE) && (currentpulse != 0)) {
printpulses();
currentpulse=0;
return;
}
}
// we didn't time out so lets stash the reading
pulses[currentpulse][0] = highpulse;
// same as above
while(! (IRpin_PIN & _BV(IRpin))) {
// pin is still LOW
lowpulse++;
delayMicroseconds(RESOLUTION);
```

```
if((lowpulse >= MAXPULSE) && (currentpulse != 0)) {
printpulses();
currentpulse=0;
return;
}
}
pulses[currentpulse][1] = lowpulse;
// we read one high-low pulse successfully, continue!
currentpulse++;
}
void printpulses(void) {
Serial.println("\n\r\n\rReceived: \n\rOFF \tON");
for(uint8_t i = 0; i < currentpulse; i++) {
  Serial.print("delayMicroseconds(");
Serial.print(pulses[i][0] * RESOLUTION, DEC);
Serial.print(");\n");
Serial.print("pulseIR(");
Serial.print(pulses[i][1] * RESOLUTION, DEC);
Serial.print(");\n");
}
}
```

APÊNDICE B – CÓDIGO FINAL PARA CONTROLE DE UMIDADE E TEMPERATURA

```
#include <LiquidCrystal.h>
#include <dht.h>
#include <SPI.h>
#include <SD.h>
#include <virtuabotixRTC.h>
#define dhtpinint A0 //Define pino do sensor DHT11 interno
#define dhtpinext A1 //Define pino do sensor DHT11 externo
#define IRledPin 53 // LED conectado no pino digital 13

int flag=0,flag30min=1,hora,minuto,segundo,dia,mes,ano;
const int chipSelect = 4; // Pino D4 ligado ao chipSelect
do cartão SD na Ethernet Shield
int led = 2; // Pino D2 ligado ao LED de indicação
long aux = 0, tempo = 0;
dht DHTi; //Inicializa o sensor 1
dht DHTe; //Inicializa o sensor 2
virtuabotixRTC myRTC(9, 8, 7); // myRTC(clock, data, rst)
LiquidCrystal lcd(22, 24, 6, 5, 3, 2);

//Array simbolo grau
byte grau[8] ={ B00001100,
                B00010010,
                B00010010,
                B00001100,
                B00000000,
                B00000000,
                B00000000,
                B00000000,};

float Tempint, Umidint, Tempext, Umidext;

void setup() {
 // pinMode(led, OUTPUT);
  Serial.begin(9600);
  pinMode(IRledPin,OUTPUT);
  lcd.begin(20,4); //Inicializa LCD
  //lcd.clear(); //Limpa o LCD
  lcd.createChar(0, grau);

  atualizaHora();
  datalog();
  Serial.println("Inicializando sistema...");
```

```
  // Realiza a checagem se o cartão de memória está no slot
da shield Ethernet
  if (!SD.begin(chipSelect)) {
  Serial.println("Cartão SD falhou!");
  return;
  }
  Serial.println("Cartão SD inicializado.");
  // Comentar o código abaixo após a primeira gravação no
Arduino (configura a hora do DS1302)
  // (segundos, minutos, hora, dia da semana, dia do mes,
mes, ano)
  myRTC.setDS1302Time(00, 15, 16, 7, 28, 11, 2015);
  Serial.println("Sistema inicializado.");
}

void loop() {

  tempo = millis();
  if(tempo - aux > 3000){
  DHTi.read11(dhtpinint);
  DHTe.read11(dhtpinext);
  Tempint = DHTi.temperature;
  Umidint = DHTi.humidity;
  Tempext = DHTe.temperature;
  Umidext = DHTe.humidity;
  aux = tempo;
  }
  lcd.setCursor(0,0);
  lcd.print("Temp_Int : ");
  lcd.print(" ");
  lcd.setCursor(10,0);
  lcd.print(Tempint,1);
  lcd.setCursor(14,0);

  lcd.write((byte)0);
  lcd.setCursor(15,0);
  lcd.print("C");

  lcd.setCursor(0,1);
  lcd.print("Umid : ");
  lcd.print(" ");
  lcd.setCursor(7,1);
  lcd.print(Umidint,1);
  lcd.setCursor(12,1);
  lcd.print("%");

  lcd.setCursor(0,2);
  lcd.print("Temp_Ext : ");
  lcd.print(" ");
  lcd.setCursor(10,2);
  lcd.print(Tempext,1);
```

```
  lcd.setCursor(14,2);

//Mostra o simbolo do grau formado pelo array
  lcd.write((byte)0);
  lcd.setCursor(15,2);
  lcd.print("C");

//Mostra o simbolo do grau quadrado

  lcd.setCursor(0,3);
  lcd.print("Umid : ");
  lcd.print(" ");
  lcd.setCursor(7,3);
  lcd.print(Umidext,1);
  lcd.setCursor(12,3);
  lcd.print("%");

  atualizaHora();

  if(Tempext > 23){
    SendChannelUpCode(); // Aumentar temperatura externa
   delay(500);
}
  //Intervalo recomendado para leitura do sensor
  //delay(2000);

//}
if (flag30min) {
  if ((segundo == 30) || (segundo == 0)) {
    datalog();
    flag30min=0;
  }
}
  else if (!flag30min) {
  if ((segundo != 30) && (segundo != 0)) {
  flag30min=1;
  }
}

}
void atualizaHora() { // Lê os dados do relógio DS1302 e os
armazena nas variáveis correspondentes
  myRTC.updateTime();
  hora = myRTC.hours;
  minuto = myRTC.minutes;
  segundo = myRTC.seconds;
  dia = myRTC.dayofmonth;
  mes = myRTC.month;
  ano = myRTC.year;
}
```

```
void datalog() {
// Realiza a leitura dos sensores e armazena esses dados
// no cartão de memória, em um arquivo denominado data.txt
  atualizaHora();
  DHTi.read11(dhtpinint);
  DHTe.read11(dhtpinext);
  Tempint = DHTi.temperature;
  Umidint = DHTi.humidity;
  Tempext = DHTe.temperature;
  Umidext = DHTe.humidity;
  File dataFile = SD.open("TESTE2.txt", FILE_WRITE);
  if (dataFile) {
  dataFile.print("Data: ");
  if (dia<10) dataFile.print("0");
  dataFile.print(dia);
  dataFile.print("/");
  if (mes<10) dataFile.print("0");
  dataFile.print(mes);
  dataFile.print("/");
  if (ano<10) dataFile.print("0");
  dataFile.print(ano);
  dataFile.print(" Hora: ");
  if (hora<10) dataFile.print("0");
  dataFile.print(hora);
  dataFile.print(":");
  if (minuto<10) dataFile.print("0");
  dataFile.print(minuto);
  dataFile.print(":");
  if (segundo<10) dataFile.print("0");
  dataFile.println(segundo);
  dataFile.print("Umidade ambiente: ");
  dataFile.print(Umidext);
  dataFile.print(" % ");
  dataFile.print("Temperatura ambiente: ");
  dataFile.print(Tempext);
  dataFile.println(" *C");
  dataFile.println("");
  dataFile.close();
  Serial.println("Dados armazenados.");
}
else {
Serial.println("Erro criando data.txt");
}
}
void pulseIR(long microsecs) {
// we'll count down from the number of microseconds
cli(); // this turns off any background interrupts
while(microsecs > 0) {
// 38 kHz is about 13 microseconds high and 13 microseconds
low
```

```
digitalWrite(IRledPin, HIGH); // this takes about 3
microseconds to happen
delayMicroseconds(10); // hang out for 10 microseconds
digitalWrite(IRledPin, LOW); // this also takes about 3
microseconds
delayMicroseconds(10); // hang out for 10 microseconds
// so 26 microseconds altogether
microsecs -= 26;
}
sei(); // this turns them back on
}
void SendChannelUpCode(){
delayMicroseconds(53008);
pulseIR(8740);
delayMicroseconds(4340);
pulseIR(560);
delayMicroseconds(540);
pulseIR(540);
delayMicroseconds(540);
pulseIR(560);
delayMicroseconds(1620);
pulseIR(560);
delayMicroseconds(540);
pulseIR(560);
delayMicroseconds(520);
pulseIR(560);
delayMicroseconds(540);
pulseIR(560);
delayMicroseconds(540);
pulseIR(540);
delayMicroseconds(560);
pulseIR(540);
delayMicroseconds(1620);
pulseIR(540);
delayMicroseconds(1640);
pulseIR(540);
delayMicroseconds(540);
pulseIR(560);
delayMicroseconds(1620);
pulseIR(560);
delayMicroseconds(1620);
pulseIR(560);
delayMicroseconds(1620);
pulseIR(560);
delayMicroseconds(1600);
pulseIR(560);
delayMicroseconds(1640);
pulseIR(540);
delayMicroseconds(540);
pulseIR(560);
delayMicroseconds(1620);
```

```
pulseIR(560);
delayMicroseconds(540);
pulseIR(540);
delayMicroseconds(540);
pulseIR(560);
delayMicroseconds(540);
pulseIR(560);
delayMicroseconds(540);
pulseIR(540);
delayMicroseconds(540);
pulseIR(540);
delayMicroseconds(560);
pulseIR(540);
delayMicroseconds(1620);
pulseIR(560);
delayMicroseconds(560);
pulseIR(540);
delayMicroseconds(1600);
pulseIR(560);
delayMicroseconds(1640);
pulseIR(540);
delayMicroseconds(1620);
pulseIR(560);
delayMicroseconds(1640);
pulseIR(540);
delayMicroseconds(1620);
pulseIR(540);
delayMicroseconds(1640);
pulseIR(540);
delayMicroseconds(38400);
pulseIR(8720);
delayMicroseconds(2200);
pulseIR(540);
delayMicroseconds(27464);
pulseIR(8740);
delayMicroseconds(2160);
pulseIR(560);
}
```