

Relatório Técnico

Blueteam

Eduard Newton Rebouças Dore – newtondore@gmail.com
Faberson Perfeito – emailpessoal2@qqrilugar.com
Felipe Maia – fe_switch@hotmail.com
Marcelo – mgdc.utfpr@gmail.com

Novembro de 2019

Resumo

Com passar do tempo a humanidade passou a conviver de forma mais prática com coisas do dia a dia incluindo a própria casa. Tendo em vista a evolução da tecnologia podemos incorporar certos costumes corriqueiros a uma máquina que possa verificar e atuar conforme a necessidade. O objetivo do projeto BlueHome é trazer o conforto, tranquilidade e segurança de uma maneira prática, com baixo custo e desempenho para sua casa agir por conta própria, evitando esquecimentos de janelas abertas em um clima chuvoso por exemplo. Porém ao mesmo tempo, pode ser customizada e comandada ao próprio gosto do usuário. O sistema então terá autonomia para decidir alguns procedimentos, como além dos citados anteriormente, também poderá acender luzes externas ao anoitecer. O aplicativo no celular disponibilizará o controle de portas, janelas e ajustes do ambiente personalizados aos proprietários. Para a segurança, um alarme com sensor de movimento, sirene e aviso via aplicativo cumprirão o seu papel.

1 Introdução

Apresentaremos neste documento o projeto BlueHome, uma POC de SmartHome utilizando componentes e tecnologias acessíveis, o qual foi documentado e desenvolvido em pouco mais de 6 semanas para a Disciplina de “Oficina de Aplicações de Dispositivos Móveis em Internet das Coisas”, ministrada pelo professor Marco Aurélio Wehrmeister, PhD, na Especialização em Desenvolvimento Móvel e Internet das Coisas de 2019-2020 da Universidade Tecnológica Federal do Paraná. Uma apresentação dos itens utilizados, as funcionalidades e o desenvolvimento do projeto será apresentado neste documento. Documentos complementares (documentação UML, cronograma, relatório de horas) serão entregues em separado.

Utilizou-se o Microcontrolador ESP32 por já conter Wi-Fi e Bluetooth embutidos. A fim de atender às expectativas do professor com relação a nosso projeto, este acabou sendo dividido em 2 Microcontroladores ESP32 com funcionalidades distintas, que se comunicavam individualmente via Wi-Fi com o dispositivo de controle (celular), e que se comunicavam de maneira unidirecional via RF para solicitar algumas funcionalidades. Houve pesquisa para a implementação das funcionalidades, porém não houve cópia de código de outras fontes. Algumas bibliotecas (tais como Blynk, Stepper, RFID e Wi-Fi) foram utilizadas para disponibilizar funcionalidades mais complexas cuja implementação não faria parte do escopo do projeto e também para viabilizar a entrega do projeto funcionando em sua íntegra no momento da apresentação.

SUMÁRIO

1	ESP32	03
1.1	PinMap	04
1.1.1	Analog-to-Digital Converter (ADC)	04
1.1.2	Digital-to-Analog Converter (DAC)	05
1.1.3	Touch Sensor	06
1.1.4	Watchdog	06
1.1.5	Interface Bluetooth v4.2 BR/EDR e Bluetooth	06
1.1.6	Boot	06
2	Reed-Switches (MEC089)	07
3	Blink	08
4	Led RGB	08
5	Relé	11
6	Motor de Passo	12
7	Leitor RFID	13
8	Sensor PIR	13
9	Sensor de Chuva	14
10	Sensor LDR	15
11	Evoluçãoes do Projeto	16
12	Características/Funcionalidades do Projeto	17
13	Características/Funcionalidades para Melhorar	21
14	Lista de Itens Utilizados na Montagem do Projeto	23
15	Desenvolvimento	25
16	Montagem da Maquete e Testes Finais	29
17	Conclusão	32
	REFERÊNCIAS	33

1 ESP32

O ESP32 é uma série de microcontroladores de baixo consumo energético em um chip integrado com micro controlador, Wi-Fi e Bluetooth. Desenvolvido por Espressif Sistemas na China com sede em Xangai, fabricado pela TSMC sucessor do micro controlador ESP8266.

De acordo com ESPRESSIF (2018), o ESP32 é definido como:

The ESP32 is a dual-core system with two Harvard Architecture Xtensa LX6 CPUs. All embedded memory, external memory and peripherals are located on the data bus and/or the instruction bus of these CPUs. With some minor exceptions (see below), the address mapping of two CPUs is symmetric, meaning that they use the same addresses to access the same memory. Multiple peripherals in the system can access embedded memory via DMA.

The two CPUs are named “PRO_CPU” and “APP_CPU” (for “protocol” and “application”), however, for most purposes the two CPUs are interchangeable. (ESPRESSIF, 2018, p.22)

Características Principais:

- Chip com Wi-Fi embutido: padrão 802.11 B/G/N, operando na faixa de 2.4 a 2.5GHz.
- Modos de operação: Client, Access Point, Station + Access Point.
- Microprocessador dual core Tensilica Xtensa 32-bit LX6.
- Clock ajustável de 80MHz até 240MHz.
- Tensão de operação: 3.3 VDC.
- Possui SRAM de 512KB.
- Possui ROM de 448KB.

- Possui memória flash externa de 32Mb (4 megabytes).
- Corrente máxima por pino é de 12mA (recomendável 6mA).
- Possui 36 GPIOs.
- GPIOs com função PWM/ 12C e SPI.
- Possui Bluetooth v4.2 BR/ EDR e BLE (Bluetooth Low Energy).

O ESP32 é um único chip acoplado com Wi-Fi padrão 802.11 operando a frequência 2.4 a 2.5 e Bluetooth e projetado com TSMC, mostrando bastante versátil e confiável.

1.1 PinMap

Referente ao PINOUT do WROOM-32 importante

Figura 1: ESP32Dev Board PINMAP

ESP32 Dev Board PINMAP									
(pu)				RESET	3.3V		GND		
SVP		ADC0			EN		GPIO23	VSPI MOSI	SPI MOSI
SVN		ADC3			GPIO 36		GPIO22		Wire SCL
		ADC6			GPIO 39		GPIO1	TX0	Serial TX
		ADC7			GPIO 34		GPIO3	RX0	Serial RX
DAC1		TOUCH9 ADC4			GPIO 35		GPIO21		Wire SDA
DAC2		TOUCH8 ADC5			GPIO32				
		TOUCH7 ADC17			GPIO33		GPIO19	VSPI MISO	SPI MISO
DAC1		ADC18			GPIO25		GPIO18	VSPI SCK	SPI SCK
DAC2		ADC19			GPIO26		GPIO5	VSPI SS	(pu) SPI SS
		TOUCH6 ADC16			GPIO27		GPIO17		
TMS	TDI	TOUCH5 ADC15	HSPI SCK	GPIO14	GPIO12		GPIO16		
(pd)			HSPI MISO		GND		GPIO4	ADC10 TOUCH0	(pd)
TCK		TOUCH4 ADC14	HSPI MOSI	GPIO13			GPIO0	BOOT ADC11 TOUCH1	(pu)
			FLASH D2	GPIO9			GPIO2	ADC12 TOUCH2	(pd)
			FLASH D3	GPIO10			GPIO15	HSPI SS ADC13 TOUCH3 TDO	(pu)
			FLASH CMD	GPIO11			GPIO8	FLASH D1	
			5V				GPIO7	FLASH D0	
							GPIO6	FLASH SCK	

Fonte: ESPRESSIF, 2019

1.1.1 Analog-to-Digital Converter (ADC)

O Esp32 integra ADCs de 12 bits e suporta medições em 18 canais (analog-enabled pins). O ULP-coprocessador no ESP32 também é projetado para medir as tensões enquanto opera em modo sleep, que permite o baixo

consumo de energia. A CPU pode ser despertada por uma configuração de limite e/ou através de outros gatilhos.

1.1.2 Digital-to-Analog Converter (DAC)

Dois canais DAC de 8 bits podem ser usados para converter dois sinais digitais em duas saídas de tensão analógica. Estes DAC duplos suportam a fonte de alimentação como referência de tensão de entrada e pode conduzir outros circuitos. Os canais duplos suportam conversões independentes.

1.1.3 Touch Sensor

O ESP32 tem 10 GPIOs de detecção capacitiva que detectam variações induzidas ao tocar ou aproximar de um GPIO com um dedo ou outros objetos. Ainda possui um sensor de Temperatura e um Sensor Hall interno, porém, para trabalhar com eles devem-se mudar as configurações dos registradores.

1.1.4 Watchdog

O ESP32 tem três temporizadores de vigilância: um em cada um dos dois módulos de temporizador (chamado o Temporizador de Watchdog Principal, ou MWDT) e um no módulo RTC (chamado RTC Watchdog Timer ou RWDT).

1.1.5 Interface Bluetooth v4.2 BR/EDR e Bluetooth LE (low energy)

O ESP32 integra um controlador de ligação Bluetooth e Bluetooth baseband, que executam os protocolos de banda base e outras rotinas de links de baixo nível, como modulação/desmodulação, processamento de pacotes, processamento de fluxo de bits, saltos de frequência, etc.

O controlador de ligação opera em três estados principais: standby, connection e sniff. Permite múltiplas conexões e outras operações, como

inquiry, page e secure simple-pairing, e, portanto, permite a Piconet e Scatternet.

1.1.6 Boot

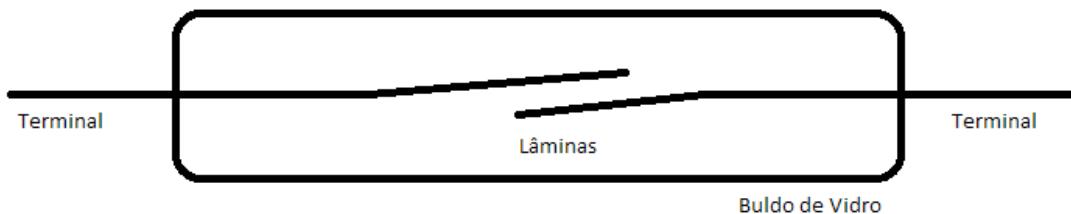
Em muitas placas de desenvolvimento com USB/Serial incorporado o esptool.py pode redefinir automaticamente a placa para o modo de inicialização. O ESP32 entrará no carregador de inicialização serial quando o GPIO0 for mantido baixo na reinicialização. Caso contrário, ele executará o programa em flash.

2 Reed-Switches (MEC089)

Segundo Braga (2019), Reed-Switches é definido como:

Os reed-switches ou interruptores de lâminas consistem em dispositivos formados por um bulbo de vidro no interior do qual existem lâminas flexíveis feitas de materiais que podem sofrer a ação de campos magnéticos. O bulbo de vidro é cheio com um gás inerte de modo a evitar a ação corrosiva do ar sobre as lâminas, o que afetaria o contato elétrico em pouco tempo.

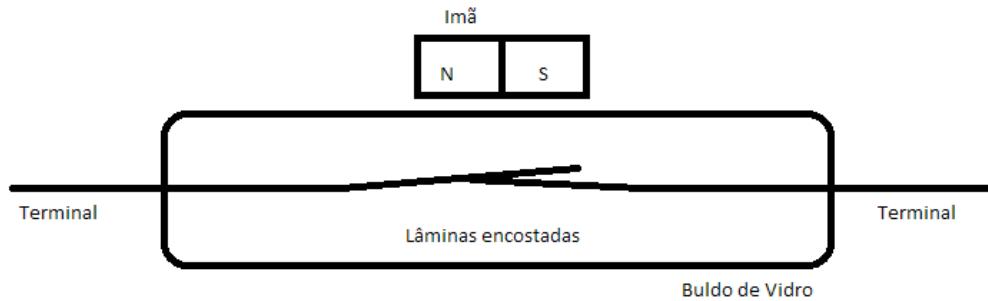
Figura 2: Reed-Switche



Fonte: Do autor (2019)

A imagem acima descreve em sua forma conceitual o Reed-Switche, temos um Bulbo de vidro que é cheio com um gás inerte de modo a evitar a ação corrosiva do ar sobre as lâminas, duas lâminas separadas e cada lâmina corresponde a um terminal. Para acionamento do dispositivo é necessário a junção das lâminas ato realizado com a aproximado um imã. Como descreve a imagem abaixo.

Figura 3: Rees-Switche Acionado



Fonte: Do autor (2019)

3 Blynk

Ao citarmos projetos com IoT (*Internet Of Things*) é quase impossível não referenciar o Arduíno, nesse contexto aumentaram a demanda de dispositivos que possuam conectividade com a internet e controle remoto destes dispositivos. Neste contexto o Blynk foi desenvolvido.

De acordo com SERRANO (2018):

Este serviço é baseado em um aplicativo personalizável que permite controlar remotamente um hardware programável, bem como reportar dados do hardware ao aplicativo. Desta forma, é possível construirmos interfaces gráficas de controle de forma rápida e intuitiva e que interage com mais de 400 placas de desenvolvimento, em sua maioria baseadas em Arduíno.

Existem três componentes principais na plataforma, *Blynk App*, *Blynk Server* e *Blynk Libraries*. O *Blynk App* permite criar interfaces de comunicação para seus projetos usando vários *widgets* padrões. *Blynk Server* responsável por todas as comunicações entre o smartphone e o hardware. *Blynk Libraries* permite a comunicação com o servidor e processam todos os comandos de entrada e saída.

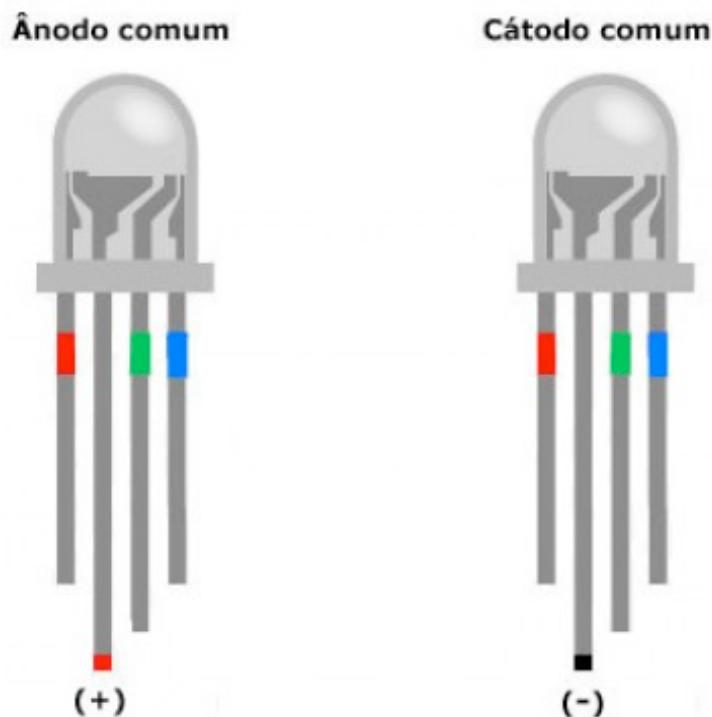
4 Led RGB

Cada letra do RGB representa uma cor primária no qual são representados por palavras no idioma Inglês:

- R de Red (vermelho)
- G de Green (verde)
- B de Blue (azul)

Com a mistura dessas cores pode-se obter até 16777216 cores, no qual podemos obter um show de luzes com apenas um led.

Figura 5: Led RGB

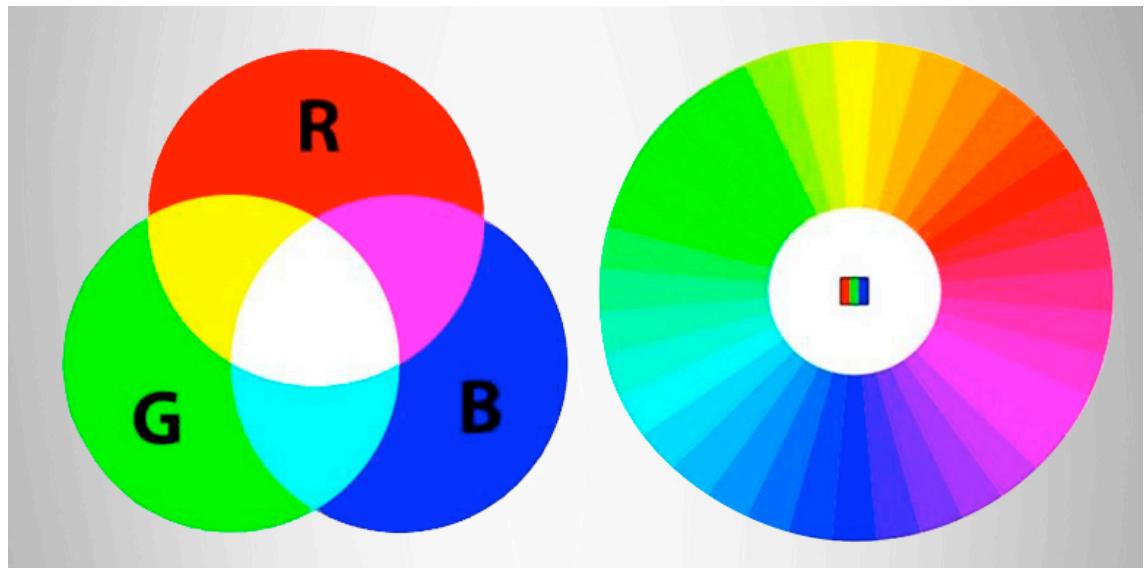


Fonte: VIDADESILICIO (2019)

A mistura aditiva destas cores nos dá a definição de cores como:

- **Branco:** A junção destas 3 cores primárias na mesma intensidade.
- **Magenta:** A junção de vermelho ao azul.
- **Amarelo:** A junção de vermelho e verde.
- **Ciano:** A junção de vermelho e azul.

Figura 6: Cores RGB



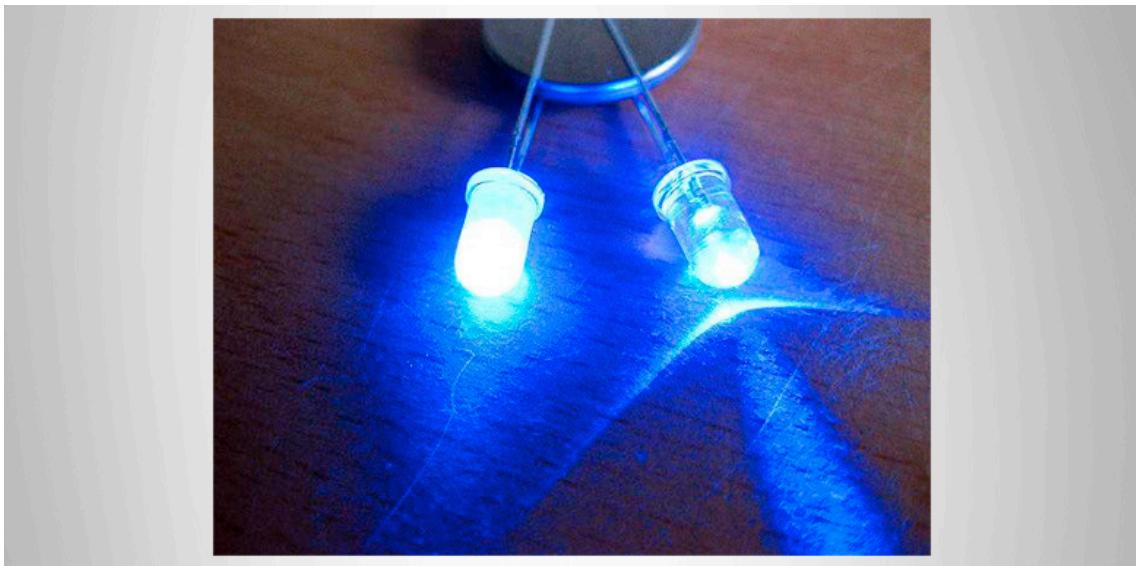
Fonte: BAUDAELETTRONICA (2019)

Um led pode ser classificado pelas suas características físicas e elétricas, tais como:

Difuso: Feixe de luz possui distribuição homogênea em todas as direções;

Alto Brilho: Feixe de luz possui maior concentração que é a mesma orientação dos terminais do LED, no sentido axial.

Figura 7: Led difuso e de alto-brilho

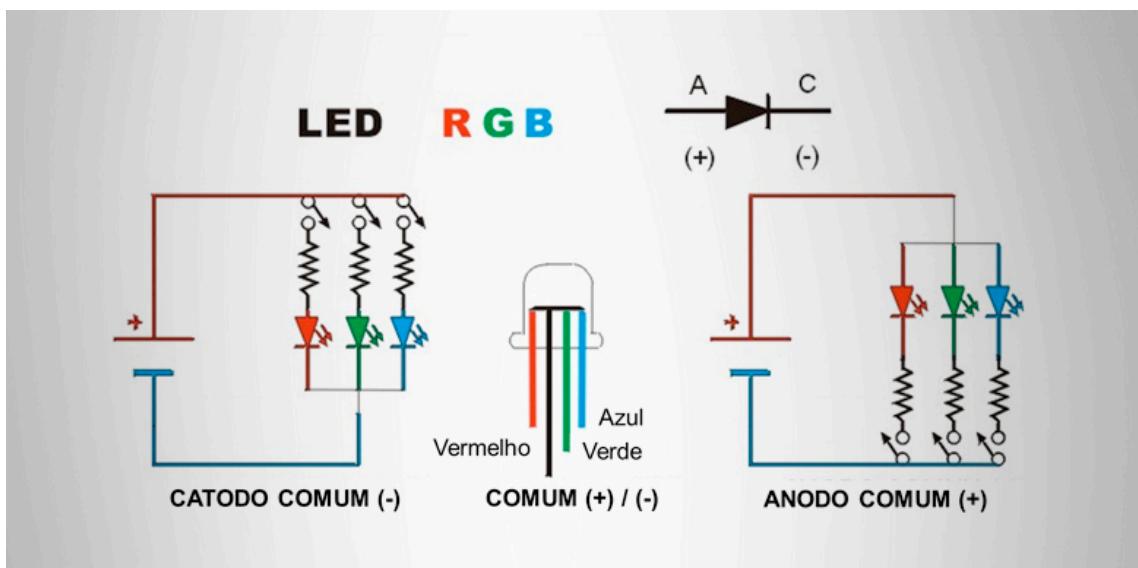


Fonte: BAUDAELETRONICA (2019)

Catodo Comum: Terminais das cores Verde, Vermelha e Azul são conectados ao terminal positivo da fonte de energia.

Anodo Comum: Terminais das cores Verde, Vermelha e Azul são conectados ao terminal negativo (GND) da fonte de energia;

Figura 8: Led catodo comum (-) e o led anodo comum (+)



Fonte: BAUDAELETRONICA (2019)

5 Relé

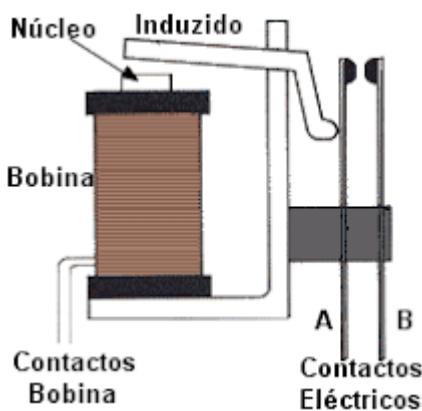
O relé é um componente eletromecânico, no qual nos possibilita acionar um interruptor a partir de um sinal. Seu sinal de comando e o interruptor são completamente isolados. Mesmo em relés pequenos o interruptor possui uma alta capacidade de corrente e tensão, os mesmos ficam dentro do relé.

O relé é muito utilizado quando precisamos acionar um dispositivo de maior potência. Os seus componentes são:

- Bobina
- Núcleo
- Induzido
- Contatos elétricos.
- Contatos da bobina

Uma bobina ao ser percorrida por uma corrente, gera um campo magnético no seu núcleo que atrai um ou vários contatos elétricos, permitindo ligar, desligar ou comutar um circuito elétrico externo.

Figura 9: Componentes do relé eletromecânico



Fonte: ELECTRONICA-PT (2019)

No exemplo da imagem, uma bobina ao receber uma tensão nos seus terminais, cria um campo magnético que através do seu núcleo atrai o induzido, fechando os contatos entre os pontos A e B.

6 Motor de Passo + Driver

Caracterizado como um motor compacto, o **motor de passo 28BYJ-48** é um motor elétrico robusto de grande utilidade, acionado por meio do **driver ULN2003**, o qual envia a corrente necessária ao motor de passo.

CARACTERÍSTICAS:

- Motor de passo;
- Controlador do motor de passo com driver ULN2003;
- 5 Vias e 4 fases;
- Possui LED's que indicam as fases acionadas;
- Encaixe rápido entre o motor de passo e o controlador;
- Rotações no sentido horário e anti-horário;
- Velocidade variável de acordo com os pulsos;
- Possui caixa de redução embutida;
- Motor compacto;
- Datasheet ULN2003A

ESPECIFICAÇÕES:

- Tensão: 5V;
- Tipo: Unipolar;
- Resistência: 60 ohms/fase;
- Torque Máximo: 2,2 Kgf.cm;
- Ângulo Passo: 5,625 x 1/64;
- Relação de redução: 1/64;
- Extensão do fio: 24cm;
- Diâmetro do motor: 28 milímetros;
- Dimensão do controlador (CxLxA): 3,5x3,1x1,1cm;
- Peso com embalagem: 41g.

7 Modulo Leitor RFID Mfrc522

O módulo leitor RFID baseado no chip MFRC522 da empresa NXP é altamente utilizado em comunicação sem contato a uma frequência de 13,56MHz. Este chip, de baixo consumo e pequeno tamanho, permite sem contato ler e escrever em cartões que seguem o padrão Mifare.

Especificações:

- Corrente de trabalho: 13-26mA / DC 3.3V
- Corrente ociosa: 10-13mA / 3.3V
- Corrente Slep: <80uA – Pico de corrente: <30mA
- Frequência de operação: 13,56MHz
- Tipos de cartões suportados: Mifare1 S50, S70 Mifare1, Mifare UltraLight, Mifare Pro, Mifare Desfire
- Temperatura de operação: -20 a 80 graus Celsius
- Temperatura ambiente: -40 a 85 graus Celsius
- Umidade relativa: 5% – 95%
- Parâmetro de Interface SPI
- Taxa de transferência: 10 Mbit/s
- Dimensões: 8,5 x 5,5 x 1,0cm
- Peso: 21g

8 Sensor PIR

O sensor PIR se caracteriza por uma alta sensibilidade e confiabilidade, sendo utilizado em inúmeros equipamentos como alarmes, luzes de emergência, acendimento automático de lâmpadas, etc.

Uma vez disparado o pino de saída quando houver movimento, o mesmo permanece ativado até que o movimento cesse.

Especificações:

- Sensor de Movimento presença PIR
- Tamanho reduzido
- Tensão de alimentação: 2.7 a 12VDC

- Consumo de energia estática: <0.1mA
- Tempo de atraso: 2 segundos
- Tempo de bloqueio: 2 segundos
- Alcance: até 5m
- Temperatura de operação: -20 a 60°C
- Dimensões: 25 x 12mm

9 Sensor de chuva

Este Sensor de Chuva pode ser usado para monitorar uma variedade de condições climáticas como gotas de chuva ou neve. Quando o clima está seco a saída do sensor fica em estado alto e quando há uma gota de chuva, a saída fica em estado baixo.

O limite entre tempo seco e chuva pode ser ajustado através do potenciômetro presente no sensor que regulará a saída digital D0. Contudo para ter uma resolução melhor é possível utilizar a saída analógica A0 e conectar a um conversor AD.

Especificações:

- Tensão de Operação: 3,3-5v
- Corrente de Saída: 100mA
- Sensibilidade ajustável via potenciômetro
- Saída Digital e Analógica
- Fácil instalação
- Led indicador para tensão
- Led indicador para saída digital
- Comparador LM393
- Dimensões Sensor de Chuva: 5x4 cm
- Dimensões Placa de Controle: 2,1x1,4 cm
- Comprimento Cabo: 20 cm

10 Sensor LDR

O Sensor de Luz LDR (Light Dependent Resistor) foi feito para detectar luz e possui uma saída digital e analógica, que podem ser conectadas diretamente em um microcontrolador como o Arduíno.

Quando a intensidade de luz está abaixo do valor ajustado, a saída do sensor fica em estado alto, e quando a intensidade de luz ultrapassa a faixa, a saída fica em estado baixo. Este valor pode ser ajustado através do potenciômetro presente no sensor que regulará a saída digital D0. Também pode-se obter informações sobre a variação de luz no sensor utilizando a saída analógica A0 do módulo.

Especificações:

- Tensão de Operação: 3-5V
- Sensibilidade ajustável via potenciômetro
- Saídas Digital e Analógica
- Fácil instalação
- Led indicador para tensão
- Led indicador para saída digital
- Comparador LM393
- Dimensões: 30 x 13mm

11 Evoluções do Projeto

O projeto, desde o momento de seu primeiro esboço, sofreu diversas modificações as quais eram sugeridas pelo professor e aceitas (como um desafio a ser alcançado) pelo desenvolvedor.

- 1 – Controle de Alarme e Automação usando um Microcontrolador ESP32.
- 2 – Controle de Alarme e Automação usando um Microcontrolador ESP32 e a inclusão de um Microcontrolador Arduíno Uno ou Mega unicamente como Multiplexador de portas (para satisfazer as portas adicionais necessárias).

3 – Controle de Alarme e Automação utilizando-se Microcontroladores ESP32 separados (aonde o Microcontrolador de Alarme também controlaria os motores da porta e da janela para deixar evidente a necessidade de comunicação entre os Microcontroladores para o funcionamento da casa), comunicando-se em contingência (em caso de falta da Wifi) por meio de porta serial com fio.

4 – Controle de Alarme e Automação separados, conforme o item 3, comunicando-se em contingência SEM fio (utilizando módulos de RF) de modo a simular uma situação mais próxima do Real, aonde os dispositivos de IoT não têm ligação física entre si.

Houve uma evolução adicional sugerida pelo professor no momento da apresentação:

5 – Separar as funcionalidades entre mais dispositivos de IoT. Isto indicaria que, no mínimo houvessem mais dois dispositivos de IoT adicionais: 1 - O controle de RFID; 2 - O controle dos motores (movimentação e informação dos sensores reed, que precisariam neste caso ser 4, indicando porta/janela totalmente fechados e totalmente abertos). Neste caso também seria possível utilizar motores normais, e com isto haveria a necessidade de uma Ponte-H para dois motores. Isto criaria a necessidade de comunicação bi-direcional entre os comunicadores de RF e também a necessidade de um protocolo seguro de comunicação entre estes, uma vez que a comunicação entre RFID e dispositivo de automação seria Wireless. Devido à limitação de alcance do Bluetooth este não seria uma solução viável para a comunicação entre dispositivos que poderiam estar fisicamente afastados por distâncias maiores do que os limites do mesmo.

12 Características e Funcionalidades Implementadas no Projeto

O projeto BlueHome, mesmo sendo uma prova de conceito, implementa as seguintes funcionalidades:

- A casa é controlada por um dispositivo de controle (celular)

O software Blynk foi utilizado para este fim, deixando mais tempo para a montagem e codificação da casa em si. A comunicação é feita via internet entre os Microcontroladores e a Nuvem do Blynk. A biblioteca Blynk do Arduíno facilitou esta implementação

- Os Processadores de Automação e Alarme são dispositivos independentes.

A comunicação entre ambos os Microcontroladores (Processadores) e o dispositivo de controle celular) é feita de maneira individual. Como o Processador de Alarme também gerencia os motores da porta e da janela, criou-se a necessidade de comunicação unidirecional entre Automação e Alarme como funcionalidade básica (que precisa funcionar mesmo sem internet) para este fim e para controle básico do alarme e da sirene. Eles utilizam módulos de RF para esta comunicação.

- A iluminação interna (cor/intensidade) pode ser definida pelo operador

O dispositivo de controle permite o ajuste da cor e da intensidade da iluminação da sala. A cor também é modificada para a cor padrão do morador por meio do TAG RFID (até sem acesso à internet) ou por meio da informação do morador (1-3 em nosso aplicativo) no dispositivo de controle.

- É possível controlar as luzes externas de maneira automática ou manual.

Um sensor de luminosidade instalado no teto da garagem informa ao processador de automação que escureceu ou clareou. Esta informação (escureceu/clareou) é passada para o dispositivo de controle. Se a configuração de iluminação automática estiver ativa, o relé da iluminação externa será chaveado automaticamente para ligado (escureceu) ou desligado (clareou), e neste caso não é possível ativar/desativar manualmente a iluminação externa. Se a iluminação automática estiver desativada, o controle da iluminação externa será feito pelo botão específico de liga/desliga a iluminação externa.

- Se chover a janela é fechada automaticamente

Se começar ou parar de chover, o dispositivo de controle é notificado. Caso a chuva comece, é verificado se a janela está fechada ou aberta. Se estiver fechada é informado no dispositivo de controle. Se estiver aberta, será feito o fechamento automático e o operador será informado. Como um complemento das funcionalidades, se a janela for aberta durante a chuva, esta não será fechada novamente até que a chuva pare e recomece. Isto permite ao morador manter a janela aberta.

- Ao passar a TAG RFID válida a casa ajusta o ambiente, abre a porta e desativa o alarme

Ao solicitar a entrada na casa por meio de uma TAG RFID válida, o ambiente da casa (atualmente é apenas a luz da sala) é ajustado conforme seus parâmetros (atualmente as Tags e as cores da iluminação estão hardcoded na programação do processador de automação), o alarme é desligado e a porta é aberta (caso esteja fechada). A autenticação positiva e o nome do morador é informada no dispositivo de controle.

Tags inválidas são notificadas por meio de som audível (3 bipes) e o dispositivo de controle é informado da tentativa. 3 tentativas inválidas ativam a Sirene mesmo que o Alarme esteja desligado, exceto se o mesmo estiver em modo Passivo (silencioso). Ao passar uma Tag válida a contagem de Tags inválidas é zerada.

- A porta e a Janela podem ser abertos/fechados pelo dispositivo de controle

Este comando é enviado diretamente do dispositivo de controle para o Processador de Alarme.

- A porta pode ser fechada a qualquer momento por um botão, mesmo sem Wi-Fi.

Existe um botão situado para efeito de demonstração junto ao RFID, porém deveria existir nas partes interna e externa da casa (preferencialmente utilizando-se de botões capacitivos) que visa fechar a porta sem a necessidade do dispositivo de controle ou de conectividade.

Uma funcionalidade adicional similar a esta poderia ser implementada para a janela, porém por ser uma POC e devido ao tempo curto, isto não foi implementado.

- O alarme pode ficar desativado, passivo (nunca toca, só avisa o dispositivo de controle) e ativo (toca se violar o perímetro)

O Alarme Inativo ainda pode ativar a sirene em alguns casos (atualmente apenas em caso de 3 tentativas consecutivas de RFIDs inválidos), e aceita a ativação/desativação manual da sirene. Estando inativo qualquer evento de invasão dos sensores (atualmente apenas o sensor PIR) não gera aviso nem no dispositivo de controle.

No modo Ativo, a invasão dos sensores informa o dispositivo de controle e dispara o alarme (dependendo do evento é preciso respeitar o tempo de espera conforme explicado no próximo item).

No modo Passivo apenas avisos são gerados. A sirene nunca é ativada (nem mesmo manualmente) durante o modo passivo.

A mudança de estado do alarme é sinalizada por um aviso sonoro: Ativo (3 bipes lentos), Passivo (3 bipes rápidos), Inativo (1 bipe longo), simulando o comportamento de um alarme real.

- A Sirene pode ser ativada ou desativada (exceto quando está em modo Passivo)

Ao desativar a sirene ou mudar a configuração do alarme ela fica em tempo de espera (como um alarme normal) por 10s (o tempo pequeno é por conta da apresentação). Durante este momento a invasão dos sensores (em nosso caso apenas o Sensor PIR) não dispara, porém o dispositivo de controle recebe o aviso.

O tempo de espera da sirene não é aplicado caso ocorram 3 tentativas inválidas consecutivas de RFID.

- Três RFIDs passadas consecutivamente ativam a sirene nos modos Ativo e Inativo

Cada tentativa inválida de Tag RFID gera um aviso na parte de automação do dispositivo de controle.

Em qualquer das 3 configurações possíveis do Alarme – Inativo, Ativo ou Passivo – três tentativas de passar RFID inválidos consecutivamente gera um aviso de Alarme no dispositivo de controle. No caso do Alarme estar configura nos modos Ativo ou Inativo a sirene tocará.

- Existe uma pequena tela de status do Processador de Automação e uma do Alarme

Esta “telinha” é representada por um LCD de 2x16 em cor azul para a automação e em cor âmbar para o alarme. Nela são apresentadas todas as mudanças de estado e ações tomadas ou executadas pelos dispositivos da casa.

13 Características/Funcionalidades do projeto que precisam ser melhoradas

Durante o desenvolvimento foi possível identificar algumas características que poderiam ou deveriam ser modificadas para tornar o projeto mais robusto e próximo do real. Elas não foram implementadas por falta de tempo para pesquisa e implementação. Mesmo não tendo feito uma análise exaustiva, foi possível identificar estas características:

- As Tags válidas e o Ambiente dos moradores está fixa no código – Precisa ser modificável via ambiente de configuração (browser) e gravada na FLASH do ESP32 de automação.
- O ambiente hoje é composto apenas de luzes da sala – Outras funcionalidades poderiam ser implementadas (temperatura padrão, playlist, entre outros), porém demandaria mais tempo. Como POC de um projeto com codificação de poucas semanas apenas a luz da sala é aceitável.
- O ambiente utiliza os dados de conectividade Wi-Fi fixos no código – Deveria ser possível passar a configuração Wi-Fi via Bluetooth ou outra maneira. Como

vários dispositivos de IoT utilizam Wi-Fi, estas configurações deveriam ser passadas via RF de um para todos os outros dispositivos.

- Dois sensores de luz em posições diferentes da casa são o ideal. Apenas em caso de escurecimento de ambos os sensores seria considerado “escuro”. Basta clarear em um sensor e o sistema informaria como claro.
- Sensores de luz e de chuva deveriam ter um número de leituras mínimas (tempo mínimo em um estado) para mudarem de estado. Isto garantiria que falsos positivos não afetariam o funcionamento
- Mais sensores de alarme (ex: micro-ondas) em complemento ao PIR poderiam garantir a leitura correta de uma invasão.
- O ambiente deveria possuir alimentação via energia da rede elétrica e via baterias na falta da primeira. A bateria deveria ser mantida carregando enquanto a energia elétrica está fora.
- Deveria ter uma maneira alternativa de alcançar o dispositivo de controle (especialmente se ele estiver próximo da casa) em caso de queda da conectividade da internet
- O dispositivo de controle deveria utilizar software próprio (deixando de usar o Blynk) ou os dispositivos da casa deveriam interagir com soluções de automação padrão da Apple ou Google.

14 Lista de Itens utilizados na montagem Projeto

Utilizou-se os seguintes itens para montar a versão final do projeto BlueHome:

- 01 Maquete em MDF de casa térrea em 1:25 para montar
- 01 Recorte em Acrílico de 50 x 100 cm
- 03 Protoboard de 400 pinos
- 03 protoboards de 170 pinos (uma ficará no módulo RFID)
- 01 Step-down ajustável com saída de 2^a (ajustado para 5.2v)
- 01 Step-down 5v-3.3v 1A
- 01 conector P4 fêmea
- 01 fonte de 9v/2A – para desenvolvimento
- 01 bateria de Li-ion 8.4V 2200mA/h (com carregador também)
- 01 chave liga/desliga
- 02 Microcontroladores ESP32
- 02 leds RGB de alto brilho transparentes
- 08 leds brancos de alto brilho
- 03 leds vermelhos de alto brilho
- 01 sirene 9V
- 01 módulo relé duplo ou 2 módulos relê individuais
- 02 buzzers 5V
- 02 Módulos de RF nRF24L01
- 01 módulo de RFID RC-522
- Resistores de 10K / 1/4w
- 01 push-button (para o fechamento da porta)
- 04 tags em formato chaveiro
- 02 motores de passo com os respectivos drivers
- 01 módulo sensor PIR
- 01 módulo sensor de chuva padrão Arduino
- 01 módulo sensor de luz compatível com Arduino
- 01 LDR grande
- Engrenagens e cremalheiras de Lego
- 40 Jumpers M/M 20 cm
- 40 Jumpers M/F 20cm
- Pinos conectores em 90º e retos
- Diversos jumpers pequenos (montados ou comprados)
- Fita dupla face para fixar os cabos de modo a não soltarem do projeto
- Diversos fios coloridos, soldas e macarrões termo retráteis de diversos tamanhos

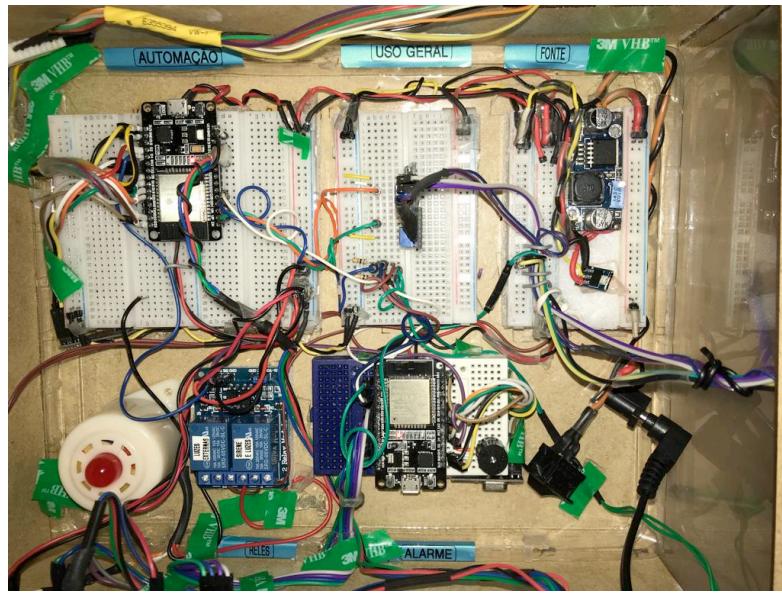


Figura D1 – A “Aranha”, o núcleo de processamento da BlueHome

15 Desenvolvimento

Conforme acertado e amplamente divulgado no início da concepção do projeto, o desenvolvimento seria feito apenas por um membro da equipe, Marcelo Costa, descrito apenas como Desenvolvedor. Outros membros cuidariam de outras necessidades (Blog, documentação, apresentações, etc), mas poderiam aproveitar tudo o quê estava sendo desenvolvido/documentado para estudarem e implementarem em seus dispositivos pessoais.

A fim de evitar qualquer necessidade de aprovação ou veto nas funcionalidades por conta do rateio de custos, todas as despesas correram por conta do Desenvolvedor. Foi adquirida uma maquete em MDF via Mercado Livre, diversos sensores/módulos, jumpers e toda a infraestrutura necessária para viabilização do projeto. Mais de 10 visitas à 24 de maio, 1 compras via ML e 2 compras em lojas on-line foram feitas para este fim.

Durante a fase de documentação inicial (1^a e 2^a semanas), o desenvolvedor estava pesquisando sobre o microcontrolador ESP32 e sobre os módulos, tecnologias e itens necessários para a viabilização da solução.

Como premissa o código teria quer ser simples o suficiente para rodar sem nenhuma ou com poucas modificações no Arduíno. Não se utilizou timers, interrupções e nem o segundo core do Processador ESP32.

Tarefas mais demoradas (ex: bipes longos e movimentação dos motores) eram divididas em tarefas menores, e executadas parcialmente a cada iteração do loop() unido as tarefas menores.

O código da automação (o mais complexo) funcionou perfeitamente e sem atrasos e com pequenos ajustes (código dos LEDS RGB foi simplificado e a conectividade foi modificada) no conjunto Arduíno Uno e ESP-01. O único detalhe, por conta do limite de memória do Arduíno Uno foi a compilação com a diretiva de #DEBUG desativada (#undef DEBUG), pois o código possui muitas mensagens de texto para depuração.

Também durante esta fase o desenvolvedor criou-se o Blog no Wix (com os primeiros posts), Facebook e Instagram para o projeto.

O desenvolvimento foi iniciado oficialmente no dia 18/10, após diversos testes isolados de funcionalidades.

No dia 19/10 o Sr Felipe Maia solicitou para participar do desenvolvimento e lhe foi incumbida a tarefa de controlar os motores de passo, com as premissas especificadas (comando de abrir, comando de fechar, sensor de abertura e sensor de fechamento). Ele recebeu um esboço inicial do código do processador de alarme e seguiu com o desenvolvimento desta funcionalidade.

A configuração do Blynk já estava quase pronta, e a tela definitiva do Blynk foi separada entre Automação (tons de azul) e Alarme (tons de âmbar) para facilitar o entendimento e a separação das funcionalidades.



Figura D2 – Tela do Blynk



Figura D3 – Outra tela do Blynk

Quanto ao Blynk, os controles da Janela e da Porta apesar de estarem em azul (automação) foram migrados para o processador de alarme. De qualquer maneira a maioria das interações da BlueHome com janela e porta partiam do processador de automação e eram passados via RF para o processador de

alarme, então a localização destes controles na parte de automação ainda parecia correta.

De qualquer maneira, conforme pode ser visto na Figura D2 o comando Fechou a Porta na automação (comando gerado pelo pressionamento do botão ao lado do leitor de RFID) gerou uma mensagem para o Processador de alarme, que iniciou o fechamento (porta fechando) e sinalizou o completo fechamento (porta fechada).

Voltando ao desenvolvimento, todas as outras funcionalidades, inclusive aquelas que o professor solicitava em sala, foram implementadas na íntegra pelo Desenvolvedor, o qual envia o código para o GIT sempre que o mesmo estava estável, permitindo que todos os integrantes pudessem baixar e testar/adaptar o código/funcionalidades em seus microcontroladores ESP-32 pessoais.

A funcionalidade de motor de passo foi baseada no código do Sr Felipe, porém as rotinas foram ajustadas para permitir a troca de direção a qualquer momento, deixando o processador ciente da quantidade exata de passos necessários para alcançar o objetivo (ex: se o processador estivesse com 30% da porta aberta e recebesse o comando de fechar, ele fecharia apenas estes 30%).

O código foi desenvolvido em ciclos de Verifica/Atua. O código do loop() ficou simples e cada uma das funcionalidades é tratada na íntegra pelo seu conjunto de verifica / atua.

```

// Rotina Principal
void loop() {
#ifndef SEMBLYNK
    Blynk.run();
#endif

    verificaStatusSensoresPortaJanela();
    processaMsgAutomacao();

    verificaAlarmeConfig();
    atuaAlarmeConfig();

    beepAsyncProcessa();

    verificaSensorPir();
    atuaSensorPir();

    verificaSireneTocando();
    atuaSireneTocando();

    verificaJanela();
    atuaJanela();
    verificaPorta();
    atuaPorta();

    delay(5);
}

```

Loop Principal:

- Ficou simples
- Só chamada a funções
- Ciclos de Verifica/Atua
- O overhead adicional de processamento é compensado pela facilidade em implementar / modificar a funcionalidade

VERIFICA:

- lê os sensores/variáveis e verifica se pode seguir com estes valores
- Corrigé (ajusta) os valores se necessário (Ex: se recebeu ordem de acender a luz externa e a mesma está em modo automático, não o faz)
- A maioria das verificações só é feita se houve mudança de estado.

ATUA:

- Executa a ação baseado nos valores
- Só atua se necessário (pendências ou mudança de estado)

Figura D4 – Loop Principal

Figura D5 – Explicações Verifica/atua

Algumas funcionalidades têm apenas verificações, pois elas são usadas em outras rotinas de verifa/atua. Ex: Os sensores de porta e janela, lidos em `verificaStatusSensoresPortaJanela()` são usados em `VerificaJanela()` e em `VerificaPorta()`.

Esta abordagem simplifica o desenvolvimento de funcionalidades por outros membros, desde que eles sigam a mesma forma de codificar (verifica/atua), porém devido ao tamanho do projeto e ao curto espaço de tempo para montar os itens eletrônicos e desenvolver o código, esta premissa não pôde ser seguida.

Antes do encerramento do dia, ou após cada funcionalidade maior implementada e com o código compilando normalmente, o projeto era enviado para o GitHub. Todas as modificações de funcionalidade das portas dos microcontroladores também eram mapeadas em um documento e atualizado no GitHub também.

Diversos links referentes a pesquisas para funcionalidades do projeto, bem como vídeos, textos e fotos eram repassados via grupo de WhatsApp para todos os integrantes durante cada fase do desenvolvimento. Desta maneira todos os integrantes, se tivessem vontade, poderiam acompanhar o desenvolvimento e aprender criando seu próprio projeto utilizando este como base e/ou tirar dúvidas.

O desenvolvimento encerrou dia 05/11, com os ajustes finais no motor de passo feitos por Marcelo e Felipe e a configuração para usar dados de celular, pois a apresentação seria num laboratório com problemas de acesso à internet.

16 A Montagem da Maquete e Testes Finais

A montagem inicial da casa, feita por todos os membros da equipe ocorreu em 28/10. Neste dia não foi possível terminar a montagem.



Figura D6 - Montagem da porta e da Janela



Figura D7 - Corte e posicionamento do teto com acrílico

Ficou óbvio que um conjunto maior de habilidades além de codificar, depurar e documentar seria necessário na montagem da maquete. Precisamos cortar, lixar, polir, colar um monte de coisas e fazer diversas outras coisas. Felizmente tinha-se todas as ferramentas necessárias para executar estas tarefas.

Por sorte tínhamos pessoas com “espírito” de Arquiteto, de Engenheiro e de Mestre de Obras ansiosos para fazer devidas adaptações.

A casa era complexa e disponha de diversos cômodos. Para deixar todo o projeto devidamente funcionando e escondido, decidimos mudar a casa para ter apenas dois cômodos: A sala (aonde seriam feitas as operações da porta e janela e a parte de trás aonde ficaria a parte eletrônica.

A montagem final da casa, aonde os motores, sensores e colagem do resto da casa aconteceu em 3 partes:

Parte 1 – Dia 03/11 - Srs Marcelo e Felipe, aonde foram fixadas as cremalheiras os motores e as engrenagens, a parte inicial da garagem e fixação dos 3 sensores (chuva, iluminação e PIR) e a placa foi colocada na casinha;

Parte 2 – Dia 04/11 – Sr Marcelo montou cabos com jumpers maiores para todos os recursos que precisariam ser desconectados, procurou itens de Lego para a parte frontal, lateral e interior da sala, e colou todos, exceto os itens do interior

Parte 3 – dia 05/11 – Srs Marcelo e Felipe montaram os extensores para o motor de passo, colaram os cabos soltos com cola quente, testaram/ajustaram o número de passos do motor, colaram os Legos do interior da sala e finalmente testaram todas as funcionalidades da BlueHome.

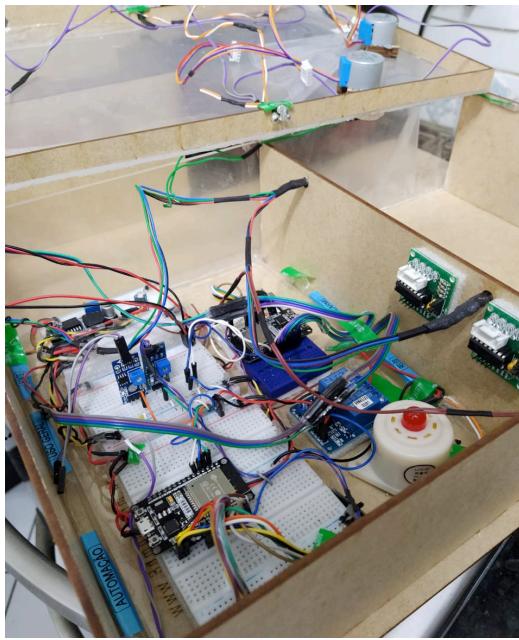


Figura D8 - Montagem final da aranha em 03/11

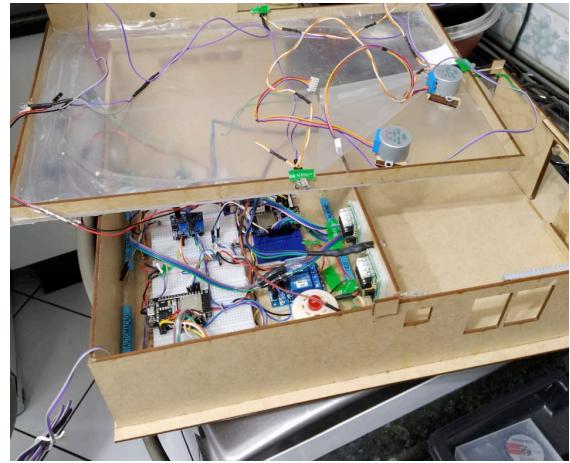


Figura D9 - Posição dos itens em 03/11



Figura D10 - Montagem final em 05/11

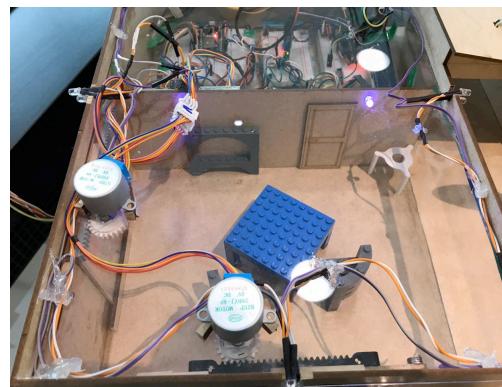


Figura D11 - parte superior em 05/11

Os outros dois membros não participaram do final da montagem da maquete por estarem trabalhando em parte da documentação técnica.

Após a montagem final restou apenas o teste dos passos exatos do motor, seguido do teste de todas as funcionalidades.

Findo isto o desenvolvimento foi encerrado e todos os itens necessários para a apresentação de 06/11 foram separados.

17 Conclusão

Apesar de ser um projeto grande e haver a necessidade de implementá-lo por completo num prazo curto, conseguiu-se entregar o mesmo com 100% das funcionalidades propostas e todas as funcionalidades adicionais solicitadas/sugeridas pelo Professor na data da Apresentação

A maior dificuldade que tivemos foi justamente manter o ânimo da equipe ao comparar o tamanho do nosso projeto (número de funcionalidades) com o dos outros apresentados na Oficina. Porém os argumentos passados pelo professor nos fizeram continuar.

Foi necessária muita pesquisa para conhecer o ESP32 e implementar as funcionalidades propostas e solicitadas. Tentou-se, sempre que possível, não utilizar funcionalidades do ESP32 que não tivessem uma contraparte no Arduíno de modo a ser possível portar a solução com facilidade. Também foram precisos muitos testes, contudo o resultado final um sucesso, o quê pôde ser visto por meio da apresentação das funcionalidades.

Por fim, a abordagem utilizada no código simplificou a implementação, pois separava a lógica de verificação/ajuste dos valores de entrada (que poderia vir através de sensores ou via dispositivo de controle) e a execução das ações relacionadas a estas entradas. Se tivéssemos mais tempo certamente implementaríamos mais funcionalidades, porém as atualmente implementadas – mesmo com algumas necessitando de melhorias – já representam um bom exemplo de Automação Residencial.

Melhorar as funcionalidades do projeto, passar a utilizar um software de controle próprio e escaloná-lo de modo a torná-lo mais próximo da realidade serão os desafios futuros que teremos.

Referência

BLINK. **Documentação**. Disponível em: <<http://docs.blynk.cc/>>. Acesso em: 30 de Out. de 2019.

BRAGA, Newton. **Como funciona o Reed-Switches**. 09 de Out de 2019. Disponivel em: <<https://www.newtoncbraga.com.br/index.php/como-funciona/3860-mec089>>. Acesso em: 22 de Out de 2019.

ESPRESSIF. Espressif Systems. **ESP32**: Technical Reference Manual. 4.0 ed. Xangai: Copyright. 2018.

ESPRESSIF. **Arduino Core For The ESP32**. 14 de Out de 2019. Disponível em: <<https://github.com/espressif/arduino-esp32>>. Acesso em: 21 de Out. de 2019.

SERRANO, Tiago Medicci; NUNEZ, Ronaldo. **Introdução ao Blink App**. 08 de Maio de 2018. Disponível em: <<https://www.embarcados.com.br/introducao-ao-blynk-app/>>. Acesso em: 30 de Out. de 2019.

K, Fernando. **Introdução ao ESP32**. 21 de Nov. de 2017. Disponível em: <<https://www.fernandok.com/2017/11/introducao-ao-esp32.html>>. Acesso em: 21 de Out de 2019.

VIDADESILICIO. **Led RGB**. 04 de Nov de 2019. Disponivel em: <<https://www.vidadesilicio.com.br/led-rgb-anodo-comum>>. Acesso em: 04 de Nov de 2019.

BAUDAELETRONICA. **Led RGB**. 04 de Nov de 2019. Disponivel em: <<http://blog.baudaelectronica.com.br/leds-rgb/>>. Acesso em: 04 de Nov de 2019.

ELECTRONICA-PT. **Relé.** 05 de Nov de 2019. Disponivel em:
<<https://www.electronica-pt.com/rele>>. Acesso em: 05 de Nov de 2019.

USINAINFO . **Motor de Passo** 08 de Nov de 2019. Disponivel em:
<<https://www.usinainfo.com.br/motor-de-passo/motor-de-passo-28byj-48-driver-uln2003-2535.html>>. Acesso em: 08 de Nov de 2019.

FILIPIFLOP. **Leitor RFID, Sensor PIR, Sensor Chuva e Sensor LDR** 08 de Nov de 2019. Disponivel em:
<<https://www.filipiflop.com>> Acesso em: 08 de Nov de 2019.

Maquete Casa Térrea 1/25 - Mdf 3mm - Kit Para Montar. Mercado Livre, 2019.
Disponível em: <https://produto.mercadolivre.com.br/MLB-712483637-maquete-casa-terrea-125-mdf-3mm-kit-para-montar-_JM>. Acesso em 09/2019

Conhecendo o ESP32. Eletrogate, 2019. Disponível em:
<<https://blog.eletrogate.com/conhecendo-o-esp32-introducao-1/>>. Acesso em 09/2019.

ESP32 Pinout Reference: Which GPIO pins should you use?. Random Nerd Tutorials, 2019. Disponível em: <<https://randomnerdtutorials.com/esp32-pinout-reference-gpios/>>. Acesso em 09/2019.

K, Fernando. ESP32: Detalhes internos e pinagem. FernandoK, 2019.
Disponível em: <<https://www.fernandok.com/2018/03/esp32-detalhes-internos-e-pinagem.html>>. Acesso em 09/2019.

ESP32 vs ESP8266 – Pros and Cons. Make Advisor, 2019.
<<https://makeradvisor.com/esp32-vs-esp8266/>>. Acesso em 09/2019.

Arduino Reference - String(). Arduino.cc, 2019. Disponível em: <<https://www.arduino.cc/reference/en/language/variables/data-types/stringobject/>>. Acesso em 09/2019.

MORAIS, José. Controle de potência via PWM – ESP32. Vida de Silício, 2019. Disponível em: <<https://portal.vidadesilicio.com.br/controle-de-potencia-via-pwm-esp32/>>. Acesso em 09/2019.

THOMSEN, Adilson. Upgrade de firmware do módulo ESP8266. FilipeFlop, 2019. Disponível em: <<https://www.filipeflop.com/blog/upgrade-de-firmware-do-modulo-esp8266/>>. Acesso em 09/2019.

Blynk - Documentação. Blynk.cc, 2019. Disponível em: <<https://docs.blynk.cc/>>. Acesso em 09/2019.

Using Arduino IDE SPI.h library with ESP32. Expressif, 2019. Disponível em: <<https://esp32.com/viewtopic.php?t=1412>>. Acesso em 09/2019.

ESP32 – Um grande aliado para o Maker IoT. FilipeFlop, 2019. Disponível em: <<https://www.filipeflop.com/blog/esp32-um-grande-aliado-para-o-maker-iot/>>. Acesso em 09/2019.

Como usar um sensor de presença com Arduino. Arduino e Cia, 2019. Disponível em: <<https://www.arduinoecia.com.br/sensor-presenca-arduino-modulo-pir-dyp-me003/>>. Acesso em 10/2019.

K, Fernando. Expansor de portas para ESP32 usando Arduíno Mega com SPI. FernandoK, 2019. Disponível em:

<<https://www.fernandok.com/2019/09/expansor-de-portas-para-esp32-usando.html>>. Acesso em 10/2019.

UTILIZANDO O BLYNK COM ESP32. Curto Circuito, 2019. Disponível em: <<https://www.curtocircuito.com.br/blog/utilizando-blynk-esp32/>>. Acesso em 10/2019.

Como usar o Módulo MP3 DFPlayer Mini. Arduino e Cia, 2019. Disponível em <https://www.arduinoecia.com.br/modulo-mp3-dfplayer-mini-dfrobot-arduino/>. Acesso em 10/2019.

K, Fernando. ESP32 com Touch Button Capacitivo. FernandoK, 2019. Disponível em <https://www.fernandok.com/2018/02/esp32-com-touch-button-capacitivo.html>. Acesso em 10/2019.

How to control a character I2C LCD with Arduino. Makerguides.com, 2019. Disponível em <https://www.makerguides.com/character-i2c-lcd-arduino-tutorial/>. Acesso em 10/2019.

nRF24L01. iMediaBank, 2019. Disponível em <http://sankios.imediabank.com/nrf24l0>. Acesso em 10/2019

Wemos ESP32 com OLED onboard. Do Bit ao Byte, 2019. Disponível em <https://www.dobitaobyte.com.br/wemos-esp32-com-oled-onboard/>. Acesso em 10/2019.

How to use ESP32 Dual Core with Arduino IDE. Random Nerd Tutorials, 2019. Disponível em <https://randomnerdtutorials.com/esp32-dual-core-arduino-ide/>. Acesso em 10/2019.

BLYNK_WRITE vs BLYNK_READ. Blynk Community, 2019. Disponível em <https://community.blynk.cc/t/blynk-write-vs-blynk-read/16613>. Acesso em 10/2019.

THOMSEN, Adilson. Controlando lâmpadas com Módulo Relé Arduino. FilipeFlop, 2019. Disponível em <https://www.filipeflop.com/blog/controle-modulo-rele-arduino/>. Acesso em 10/2019.

Cooler com Arduino e pastilha termoelétrica TEC1-12706. Arduino e Cia, 2019. Disponível em <https://www.arduinoecia.com.br/cooler-arduino-pastilha-termoeletrica-tec1-12706/>. Acesso em 10/2019.

ESP32 com RFID: Controle de Acesso. FernandoK, 2019. Disponível em <https://www.fernandok.com/2018/02/esp32-com-rfid-controle-de-acesso.html>. Acesso em 10/2019.

Guide for WS2812B Addressable RGB LED Strip with Arduino. Random Nerd Tutorials, 2019. Disponível em <https://randomnerdtutorials.com/guide-for-ws2812b-addressable-rgb-led-strip-with-arduino/>. Acessado em 10/2019.

ESP8266 Publishing DHT22 Readings with MQTT to Raspberry Pi. Random Nerd Tutorials, 2019. Disponível em <https://randomnerdtutorials.com/esp8266-publishing-dht22-readings-with-mqtt-to-raspberry-pi/>. Acessado em 10/2019.

Getting Started with Raspberry Pi 3. Random Nerd Tutorials, 2019. Disponível em <https://randomnerdtutorials.com/getting-started-with-raspberry-pi/>. Acessado em 10/2019.

How 433MHz RF Tx-Rx Modules Work & Interface with Arduino. Last Minute Engineers, 2019. Disponível em <https://lastminuteengineers.com/433mhz-rf-wireless-arduino-tutorial/>. Acesso em 10/2019.

Três maneiras de configurar a segunda serial do ESP32. Do Bit ao Byte, 2019. <https://www.dobitaobyte.com.br/tres-maneiras-de-configurar-a-segunda-serial-do-esp32/>. Acesso em 10/2019.

MOXHAM, James. Arduino String Manipulation Using Minimal Ram. Instructables Circuits, 2019. Disponível em <https://www.instructables.com/id/Arduino-String-Manipulation-Using-Minimal-Ram/>. Acesso em 10/2019.

MCCAULEY, Mike.AccelStepper library for Arduino. AirSpayce, 2019 - Disponível em <http://www.airspayce.com/mikem/arduino/AccelStepper/>. Acesso em 11/2019.

Stepper - Reference. Arduino, 2019. Disponível em <https://www.arduino.cc/en/reference/stepper>. Acesso em 11/2019.