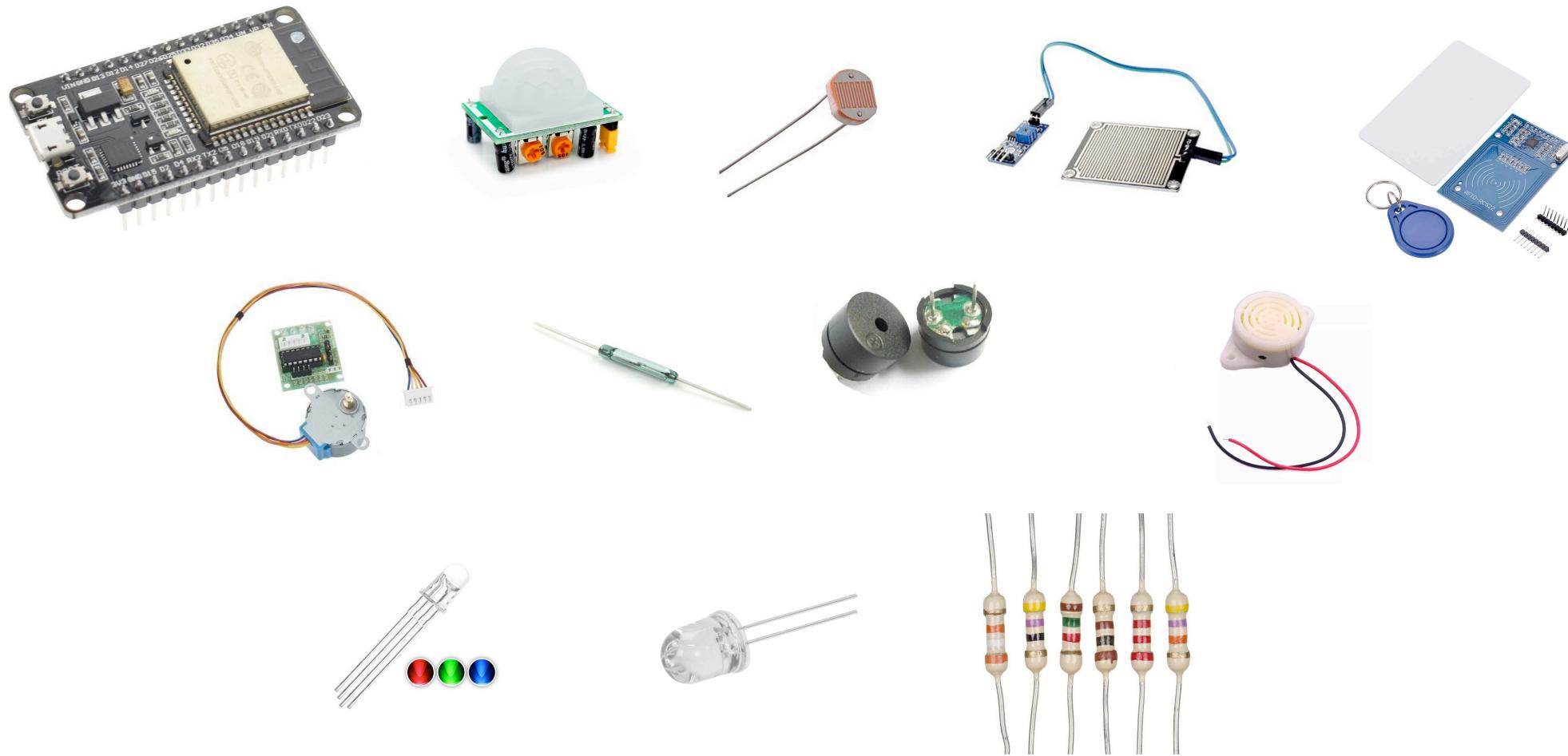


BlueHome

Uma solução para a automatização residencial, visando a facilidade e o baixo custo com qualidade e desempenho. Usando dispositivos de Internet da Coisas, sensores infravermelhos e de luminosidade, esp 32, RFID e aplicativo mobile para trazer conforto e tecnologia à casa. Trata-se de uma prova de conceito, sendo utilizada uma maquete para tal comprovação.

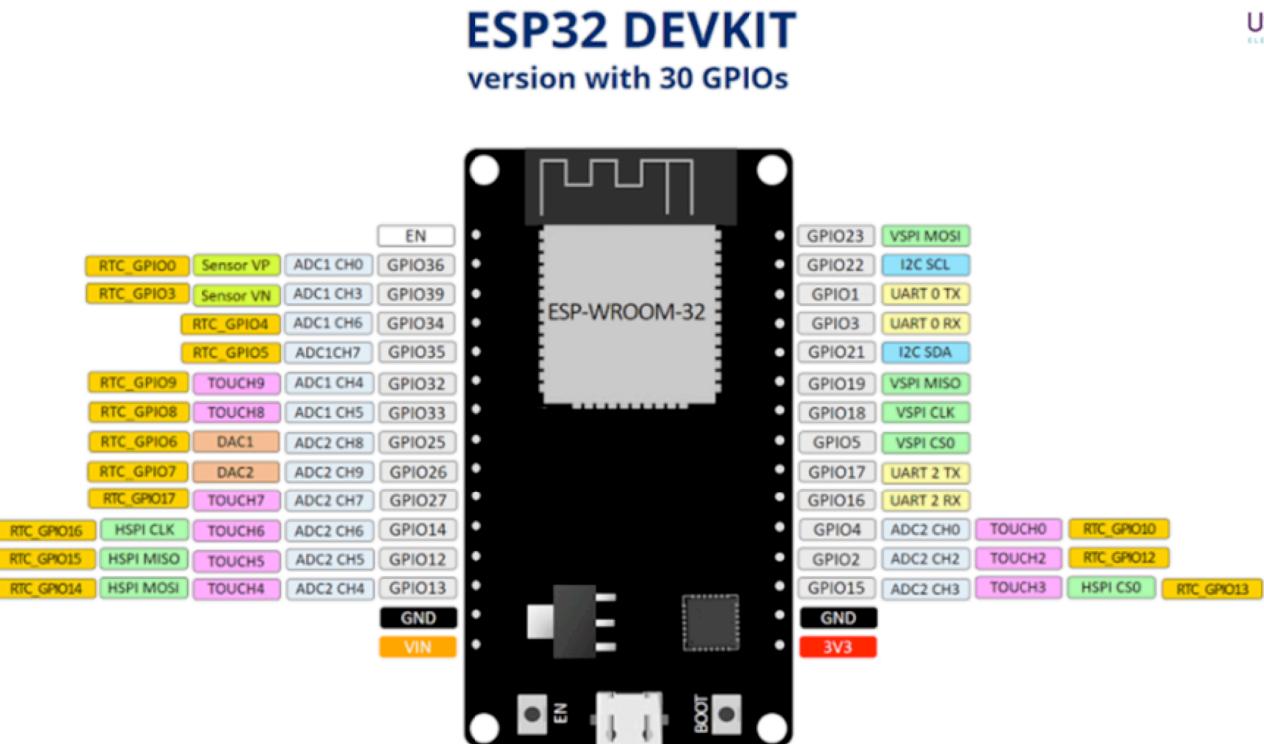
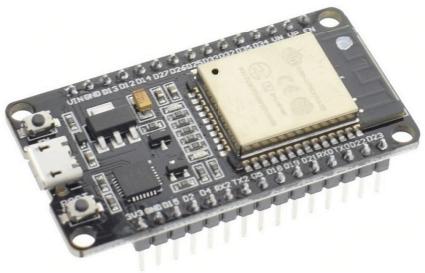
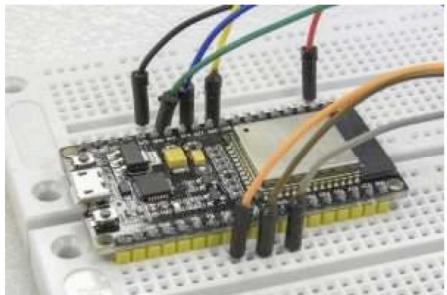


Materiais Utilizados



Esp32

Microcontrolador de baixo custo e baixo consumo de energia. O processador possui 2 cores em alta velocidade, um core em baixa velocidade (low-power), Com Wi-Fi e Bluetooth integrados.



Maquete:

Escala: 1:25 - Material: Madeira MDF 3mm - Dimensões: 48x40x17 CxLxA Originalmente.



Arduino IDE

É usado para escrever e fazer upload de programas em placas de microprocessadores com Arduino, esp32, entre outros

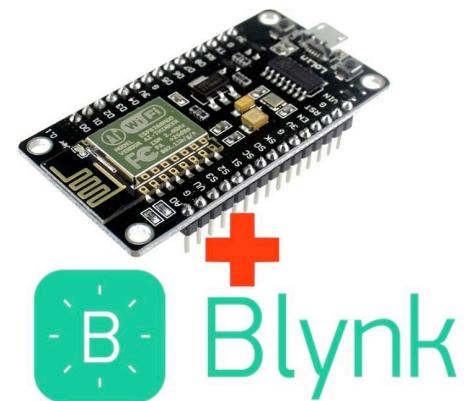
The image shows two side-by-side windows of the Arduino IDE. Both windows have the title "DHT11_humidity | Arduino 1.6.12" and "DHT11_humidity | Arduino 1.0.6". The code in both windows is identical, reading:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
#define DHT11_PIN 0 // ADC0
byte read_dht11_dat()
{
    byte i = 0;
    byte result=0;
    for(i=0; i< 8; i++){
        while(!(PINC & _BV(DHT11_PIN))); // wait for 50us
        delayMicroseconds(30);
        if(PINC & _BV(DHT11_PIN))
            result |=(1<<(7-i));
    }
}
```

The bottom status bar indicates "NodeMCU 1.0 (ESP-12E Module), 80 MHz, 256000, 4M (3M SPIFFS) on COM15" for the left window and "Arduino Mega (ATmega1280) on" for the right window.

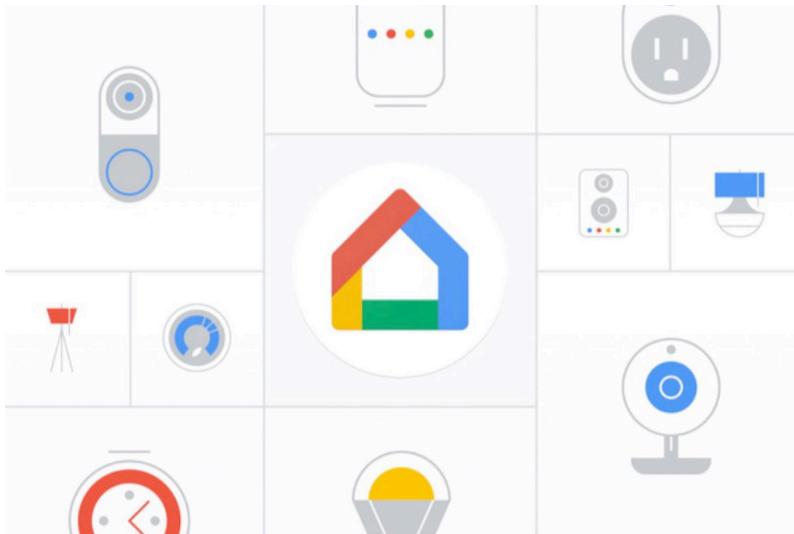
BLYNK

É uma empresa de tecnologia que desenvolve infraestrutura para a Internet das Coisas

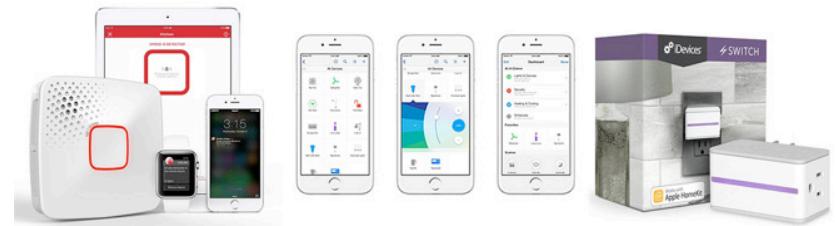


Outras Soluções

Google Home: Só caixa de som R\$349,90. App gratuito



Apple Home Kit: R\$600 a R\$7.000 depende do numero de componentes. App gratuito

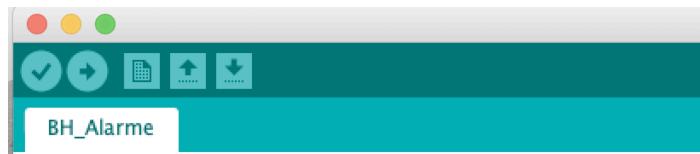


Como foi desenvolvido

- Código estruturado nos métodos verifica(), atua().
- Comunicação entre os microcontroladores via RF.
- Não foram utilizados timers ou interrupções.
- Portável de maneira simples para outros sistemas
- Tarefas maiores foram divididas em micro-taferas, por exemplo: um giro do motor de passo: um a cada execução do loop()
- Tarefas essenciais não dependem de rede wifi ou conexão com o dispositivo de controle para o funcionamento

Loop principal

- Fácil de entender
 - Só chamadas a funções
 - Cada chamada com uma funcionalidade
 - Ciclos de Verifica / Atua
-
- VERIFICA:
 - lê os sensores/variáveis e verifica se pode seguir com estes valores
 - Corrige (ajusta) se necessário
 - ATUA:
 - Executa a ação baseado nos valores



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

    verificaStatusSensoresPortaJanela();
    processaMsgAutomacao();

    verificaAlarmeConfig();
    atuaAlarmeConfig();

    beepAsyncProcessa();

    verificaSensorPir();
    atuaSensorPir();

    verificaSireneTocando();
    atuaSireneTocando();

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

    delay(5);
}
```

Exemplo de um “Verifica” simples

```
void verificaStatusSensoresPortaJanela () {  
    bool lidoPortaFechada = (digitalRead(PIN_SENSORPORTA)==LOW);  
    bool lidoJanelaFechada = (digitalRead(PIN_SENSORJANELA)==LOW);  
  
    if (_isPortaFechada!=lidoPortaFechada) {  
        _isPortaFechada=lidoPortaFechada;  
  
        #ifdef DEBUG  
            Serial.println("- Porta Fechada Mudou de Status para "+String(_isPortaFechada));  
        #endif  
    }  
  
    if (_isJanelaFechada!=lidoJanelaFechada) {  
        _isJanelaFechada=lidoJanelaFechada;  
  
        #ifdef DEBUG  
            Serial.println("- Janela Fechada Mudou de Status para "+String(_isJanelaFechada));  
        #endif  
    }  
}
```

Exemplo de um “Atua” mais elaborado

```
// Atua na Configuração do Alarme (se necessário)
void atuaAlarmeConfig () {

    if (_atualAlarmeConfig!=_novoAlarmeConfig) {

        // Se ativará o alarme agora (ativo ou passivo), registra o tempo em que os sensores Não Dispararão a Sirene
        if (_atualAlarmeConfig==ALARME_CONFIGINATIVO) {
            _sireneInativaPorSensorAte=millis()+ALARME_TEMPOESPERA;
        }

        _atualAlarmeConfig=_novoAlarmeConfig;

        switch (_atualAlarmeConfig) {
            case ALARME_CONFIGINATIVO : beepAsyncPlay(BEEP_ALARME_INATIVO);
                printLCD("Alarme Desligado");
                _novoSireneTocando=false;
                break;
            case ALARME_CONFIGATIVO   : beepAsyncPlay(BEEP_ALARME_ATIVO);
                printLCD("Alarme ATIVO");
                break;
            default :
                beepAsyncPlay(BEEP_ALARME_PASSIVO);
                _novoSireneTocando=false;
                printLCD("Alarme PASSIVO");
        }
        // Marca o tempo de espera

        Blynk.virtualWrite(BLYNK_ALARMECONFIG,_atualAlarmeConfig);
#ifdef DEBUG
        Serial.println("-- Mudou a Configuração do Alarme: "+String(_atualAlarmeConfig));
#endif
    }
}
```

Exemplo do Verifica/Atua da Sirene Tocando

```
// Verifica/Valida a condição do alarme tocando se ela mudou
void verificaSireneTocando () {
    // Premissa: Se estiver em Modo Passivo, NUNCA pode deixar o alarme tocar (se estiver desligado, 3 RFIDs inválid
    if (_atualAlarmeConfig==ALARME_CONFIGPASSIVO) && (_novoSireneTocando)) {
        _novoSireneTocando=false;
        // Precisa avisar o Blynk, pois a mudança de estado pode ter ocorrido por lá
        Blynk.virtualWrite(BLYNK_SIRENETOCANDO,LOW);
        // APENAS Avisa no LCD que houve violação do Alarme
        printLCD("ALARM.SILENCIOSO");
    }
}

// Atua na Sirene do Alarme (se necessário)
void atuaSireneTocando () {
    if (_atualSireneTocando!=_novoSireneTocando) {
        _atualSireneTocando=_novoSireneTocando;

        Blynk.virtualWrite(BLYNK_SIRENETOCANDO,_BoolToEstado(_atualSireneTocando));
        digitalWrite(PIN_RELALARME,!_BoolToEstado(_atualSireneTocando));

        if (_atualSireneTocando)
            printLCD("*SIRENE ATIVA*");
        else {
            printLCD("*Sirene Inativa*");
            // Não permite que a sirene 'arme' de novo'por um certo momento...
            _sireneInativaPorSensorAte=millis()+ALARME_TEMPOESPERA;
        }
    }

    #ifdef DEBUG
        Serial.println("-- Mudou o Status de Sirene Tocando: "+String(_atualSireneTocando));
    #endif
}
```

Verifica Motor da Porta

BH_Alarme

```
// Verifica se mudou de "direção" o movimento da Porta e ajusta o ambiente se necessário
void verificaPorta () {
    if (_atualPortaDirecao!=_novoPortaDirecao) {
        _atualPortaDirecao=_novoPortaDirecao;

        printLCD("Porta "+_getStrDirecao(_atualPortaDirecao));
        _isPortaMovendo=true;

        Blynk.virtualWrite(BLYNK_PORTADIRECAO,_atualPortaDirecao);
        //    printLCD("Porta "+_getStrDirecao(_atualPortaDirecao));
#ifdef DEBUG
        Serial.println("-- Mudou a Direcao da Porta: "+_getStrDirecao(_atualPortaDirecao));
#endif
    }
}
```

Atua no Motor da Porta

BH_Alarme

```
void atuaPorta () {
    if (_isPortaMovendo) {
        if (_atualPortaDirecao>0) {
            if (_portaPosicao>MOTORPORTA_POSFECHADA) {
                _portaPosicao++;
                motorPorta.step(MOTORPORTA_PASSOSVEZ);
            }

            // Se chegou ao fim, para...
            if (_portaPosicao>=MOTORPORTA_POSFECHADA) {
                _isPortaMovendo=false;
            }
        } else {
            if (_portaPosicao>MOTORPORTA_POSABERTA) {
                _portaPosicao--;
                motorPorta.step(-MOTORPORTA_PASSOSVEZ);
            }

            // Se chegou ao fim, para...
            if (_portaPosicao<=MOTORPORTA_POSABERTA) {
                _isPortaMovendo=false;
            }
        }
    }
#endif DEBUG
    Serial.println(" - Moveu Porta. Posicao Atual = "+String(_portaPosicao));
#endif
    // Se não está mais movendo, acabou...

    if (!_isPortaMovendo) {
        printLCD("Porta "+_getStrDirecaoFim(_atualPortaDirecao));
#endif DEBUG
        Serial.println("--Porta OK. Status = "+_getStrDirecaoFim(_atualPortaDirecao));
#endif
    }
}
```

Por quê usar esta abordagem ?

- Separa as lógicas de maneira clara e simples
- Após entender o conceito, torna-se fácil de implementar
- É fácil de adicionar novas funcionalidades ou melhorar alguma já existente
- Pessoas diferentes podem desenvolver as partes individuais do código e juntar depois

NOSSAS FUNCIONALIDADES

Querem saber por aqui
ou apenas na demonstração ???

NOSSAS FUNCIONALIDADES

- A casa é controlada por um dispositivo de controle (celular)
- Os processadores de Automação e Alarme são dispositivos independentes. Eles se comunicam sem fio
A iluminação interna (cor/intensidade) pode ser definida pelo operador.
- Se chover ele fecha a janela e avisa operador.
Se abrir durante a chuva a janela ficará aberta até parar e começar a chover novamente
- Ao passar a TAG RFID válida a casa ajusta o ambiente (cor da luz da sala) e abre a porta.
Passando 3 vezes uma TAG inválida toca a sirene e alerta operador (exceção: se alarme estiver em modo passivo apenas avisa)
- A porta e a Janela podem ser abertos/fechados pelo celular. A porta pode ser fechada por um botão também
- Pode controlar as luzes externas de maneira automática ou manual. Avisa o operador se escureceu/clareou
- O alarme pode ficar desativado (toca apenas no caso de RFID inválido), passivo (nunca toca, só avisa) e ativo (toca se violar o perímetro ou por RFID)
- A Sirene pode ser ativada ou desativada (exceto quando está em modo Passivo). Ao desativar a sirene ela fica em tempo de espera (como um alarme normal) por 10s (*para dar tempo de sair*)
- Existe uma pequena tela de status do Processador de Automação e uma do Alarme.

A “Aranha” - Nosso Projeto

