

Automação residencial: Comandando lâmpadas pelo *Telegram*

Parley Martins - 11/0038096 Tatielen Pereira - 12/0136074

Resumo—Este trabalho propõe a utilização de automação residencial para possibilitar ao usuário acender e apagar lâmpadas remotamente, utilizando integração com aplicativo no celular.

Index Terms—Automação Residencial, Telegram, bot, *smart house*

1 INTRODUÇÃO

Automação residencial é resultado da combinação de espaços residenciais, como sala, banheiro, quarto com tecnologias, para maior conforto, segurança, ou menos contato humano [1]. Estas tecnologias e ideias eram, até recentemente, consideradas sonhos de um futuro distante [2], sem uso prático, exceto no entretenimento.

No entanto com um mundo conectado pela internet, que mudou o jeito que as pessoas se comunicam e se relacionam, é normal que este conceito esteja cada vez mais próximo da realidade das pessoas. Para ter mais conforto, já é possível controlar pelo celular o volume das televisões (e outros aparelhos de som), o canal em que se está, a intensidade com que aparelhos devem funcionar, entre outras comodidades. Para ter mais segurança, é possível controlar luzes, sistema de alarmes, de detecção de movimentos, etc. Existem diversas empresas que fornecem esse tipo de serviço, mas eles ainda podem ter um custo muito elevado.

2 SOLUÇÃO

Para facilitar e desmistificar o acesso à automação residencial, a proposta deste projeto é implementar um sistema que possa controlar remotamente as lâmpadas de uma casa. O

usuário, após instalação do sistema físico, poderá utilizar seu *smartphone* para ligar e desligar as lâmpadas.

A interação com o usuário se dará através de um bot no aplicativo *Telegram*. Deve-se iniciar uma 'conversa' com o bot, e mandar o comando desejado (ligar ou desligar, por exemplo). Este irá mandar para o módulo wifi do sistema, que fará a comunicação com o MSP, desligando ou ligando a lâmpada selecionada.

Para fins deste trabalho, uma lâmpada e uma fonte de energia externas, controladas pela protoboard, serão utilizadas para facilitar a instalação e testes.

O *hardware* será composto, inicialmente, pelos seguintes itens:

- protoboard, para execução do sistema;
- microcontrolador MSP430, irá executar o controle da energia na lâmpada;
- módulo esp8266, proverá o acesso à rede wifi;
- lâmpada, para testes;
- fonte de energia, tanto para o microcontrolador quanto para a lâmpada.

O *software* embarcado no microcontrolador será escrito nas linguagens C e Assembly, enquanto o código do *bot* será desenvolvido utilizando Python 3. Os serviços serão conectados através do IFTTT, que conecta servidores de terceiros a outros serviços [3].

2.1 Requisitos

O *software* do microcontrolador deve corretamente identificar os comandos e apagar ou acender a lâmpada, conforme instrução recebida.

O *bot*, *software* que responde a comandos pré definidos automaticamente, deve ser integrado ao aplicativo *Telegram* e deve mandar instruções de ligar e de desligar a lâmpada.

O sistema completo, tanto hardware quanto *software*, deve ter acesso à internet para o funcionamento correto.

2.2 Benefícios

Este projeto tem como principal beneficiário o cidadão comum que quer ter um pouco do conforto que a automação residencial traz a sua casa. Além disso, ajudará na economia de energia, já que a pessoa pode mandar um comando de apagar determinada luz, mesmo a distância.

3 DESENVOLVIMENTO

Esta seção apresenta um detalhamento do sistema proposto no projeto. Para isto, será apresentada a descrição do hardware e software.

3.1 Hardware

O Hardware utilizado pelo grupo consiste nos itens descritos na Tabela 1

Tabela 1
Tabela de componentes do projeto

Componentes	Quantidade
MSP430G2553	1
Módulo Wireless ESP8266	1
Protoboard	1
Relé	1
Diodo	1
Lâmpada Led 12w	1
Módulo relé	1
USB to TTL D-Sun (conversor USB-serial)	1
Jumpers	Indefinido

O diagrama de blocos tem como objetivo apresentar o princípio de funcionamento do sistema. Os blocos apresentados de acordo com a Figura 1 são:

- Interface de apresentação será o meio de comunicação entre o usuário e o sistema.

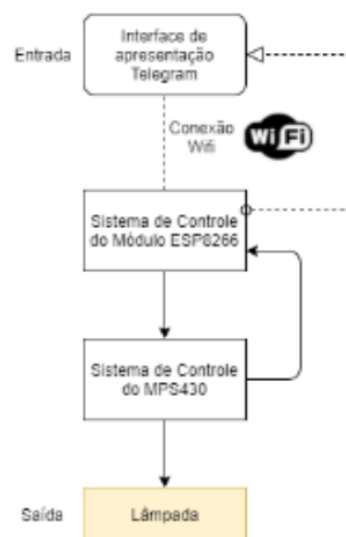


Figura 1. Diagrama de blocos do projeto

Com o auxílio do aplicativo Telegram será possível escolher se o usuário acenderá, apagará ou verificar o estado em que a lâmpada se encontra.

- Sistema de controle do módulo ESP8266 Recebe o sinal do *bot* do Telegram e manda para o MSP a entrada correspondente. Retorna ao telegram com alguma mensagem útil para o usuário saber o status da sua requisição. Como o ESP não funcionou de modo esperado, esse sistema (servidor) foi simulado no próprio computador.
- Sistema de controle do MSP430 será ligado junto ao módulo Wifi, onde será recebida e executada a decisão do usuário. Como o usuário possui a possibilidade de verificar o estado da lâmpada sistema de controle do MSP430 também envia um sinal de retorno para o módulo Wifi.
- Lâmpada será acesa ou apagada dependendo da decisão do usuário.

Para conectar o módulo Wifi ESP8266 ao MSP430 foram conectados os seguintes pinos, de acordo com a Tabela 2. A lâmpada até o momento só foi testada com o LED 1 do próprio MSP.

Tabela 2
Tabela de conexões MSP-ESP

Módulo Wifi ESP8266	MSP430
TX	P1.4
CH_PD	VCC
RST	
VCC	VCC
GND	GND
GPIO2	
GPIO0	
RX	P1.3

Não foi possível fazer a conexão entre o MSP430 e o ESP8266. Então foi feita uma conexão entre o conversor USB-serial e o MSP430 utilizando um servidor simulado no computador, que será explicado no próximo tópico. Para ligar a lâmpada ao MSP foi utilizado um módulo relé 5V.

3.2 Software

O MSP foi configurado para receber um caractere na conexão UART que troca ou lê o estado da lâmpada. Os caracteres aceitos são *n*, para ligar; *f*, para desligar; e *s* para ler e retornar o estado atual da lâmpada. Qualquer outro input resultará em nenhum retorno ou ação do MSP. O código 1 do Apêndice A inicializa o modo de comunicação UART com *baud rate* 9600 e *clock* em 1MHz. Caso algum dado seja recebido, uma interrupção do RX do UART será acionada. Esta interrupção lerá o valor recebido e tomará uma ação de acordo com o explicado acima. Para substituir a lâmpada, por enquanto, o LED conectado ao pino P1.6 do MSP está sendo utilizado.

O *bot* do Telegram foi feito para responder à comandos pré-definidos. Após o uso de um deles, o *bot* responde o usuário com uma mensagem de texto ou emoji, para demonstrar que o MSP recebeu e executou o comando, como demonstrado na Figura 2.

- */on*, vai ligar a lâmpada, retornando um emoji de lâmpada acesa.
- */off* desliga a lâmpada e retorna uma lua nova (pra demonstrar que a lâmpada foi apagada).
- */state* retorna o estado do lâmpada, com a mensagem "Light is on/off".

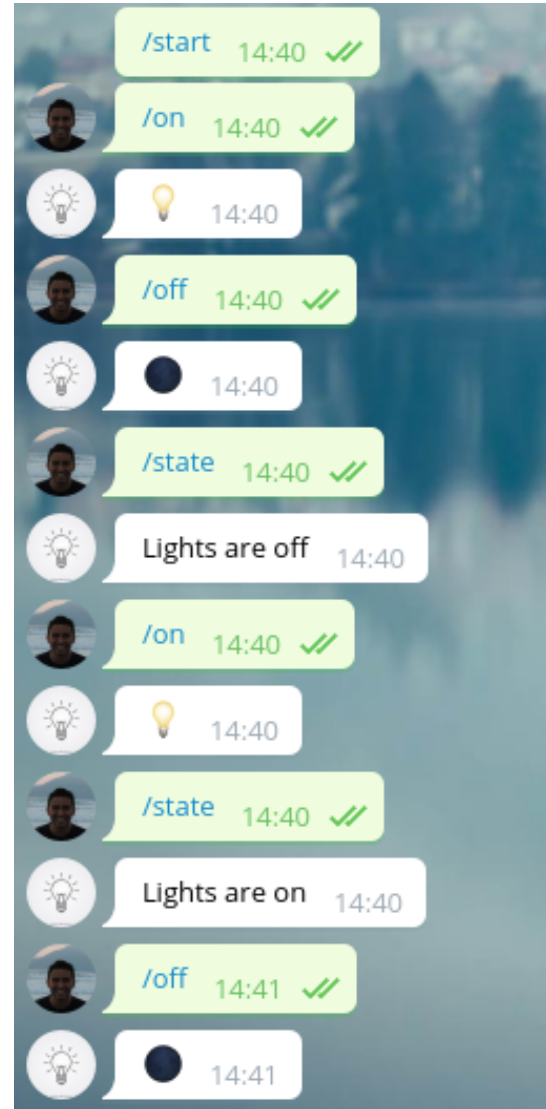


Figura 2. Comandos do Telegram

Para o Código 2 do *bot* a biblioteca *python-telegram-bot*, que é um *wrapper* da API do Telegram para fazer bots escrito em Python, foi utilizada. Primeiramente alguns *imports* são feitos e o log é definido. A função *send_message* recebe um caractere que será enviado ao MSP. A conexão serial é aberta, o caractere é enviado e a conexão é fechada. A função *read_return_message* faz o mesmo da anterior, mas lê o retorno da mensagem mandada. As três próximas funções, *turn_on*, *turn_off* e *check_state* são para gerenciar os comandos do *bot*. Elas mandam uma mensagem para o MSP e depois uma para o usuário com o retorno da comunicação serial ou uma mensagem indicando que a ação foi executada.

Os comandos são adicionados ao *dispatcher* do *bot*.

4 RESULTADOS

Inicialmente foi desenvolvido o código referente a proposta do projeto que seria de ligar uma lâmpada via Telegram por uma conexão Wi-fi utilizando modulo wireless. No entanto, não foi possível obter a conexão entre o modulo wireless e o MSP430, mesmo após a pesquisa de várias formas para obter sucesso na conexão.

Para fins didáticos, o objetivo do trabalho foi alcançado, afinal há o controle da lâmpada pelo Telegram com êxito. No entanto, porque o ESP não funcionou como esperado, o projeto ficou um pouco inviável, sendo apenas um protótipo funcional para trabalhos futuros.

5 CONCLUSÃO

A realização deste projeto foi agradável, oferecendo oportunidade de aprendizagem e construção de novos conhecimentos ainda não explorados pela dupla de alunos. Muitas dificuldades foram encontradas durante a construção do projeto, principalmente a conexão entre o módulo Wifi e o MSP.

Apesar do objetivo geral deste projeto não ter sido totalmente alcançado, os conhecimentos adquiridos durante a execução do projeto foram importantes para consolidar o aprendizado das aulas expositivas realizadas, bem como adquirir experiência prática na construção de circuitos e desenvolvimento de código em microcontroladores.

APÊNDICE A - CÓDIGO MSP

Listing 1. lamp.c

```
1 #include <msp430g2553.h>
2 #include <legacymsp430.h>
3
4 #define RX BIT1
5 #define TX BIT2
6 #define LAMP BIT6
7
8 void send_data(unsigned char c);
9 void init_uart();
10
11 int main(void)
12 {
```

```
13 WDTCTL = WDTPW + WDTHOLD;
14
15 BCSCCTL1 = CALBC1_1MHZ;
16 DCOCTL = CALDCO_1MHZ;
17
18 P1OUT &= ~LAMP;
19 P1DIR |= LAMP;
20
21 init_uart();
22 _BIS_SR(GIE);
23 return 0;
24 }
25
26 void send_data(unsigned char c)
27 {
28     while((IFG2&UCA0TXIFG)==0);
29     UCA0TXBUF = c;
30 }
31
32 void init_uart()
33 {
34     P1SEL2 = P1SEL = RX + TX;
35     UCA0CTL0 = 0; //UART, 8bits, no parity,
36     UCA0CTL1 = UCSSEL_2; // SMCLK
37     UCA0BR0 = 104; //Baud rate: 9600
38     UCA0BR1 = 0;
39     UCA0MCTL = UCBRF_0 + UCBRS_1; //Baud rate:
40         9600
41     IE2 |= UCA0RXIE; // Set interrruption by
42         UART data arrival
43 }
44
45 void send_state(char state[]){
46     int i = 0;
47
48     for(i = 0; state[i] != '\0'; i++){
49         send_data(state[i]);
50     }
51     send_data('\n');
52 }
53
54 interrupt(USCIAB0RX_VECTOR) set_lamp_state(
55     void){
56     unsigned char state = UCA0RXBUF;
57     if(state == 'n'){ // turn the light on
58         P1OUT |= LAMP;
59     } else if(state == 'f') { // turn the light
60         off
61         P1OUT &= ~LAMP;
62     } else if(state == 's') { // check light
63         state
64         if((P1OUT&LAMP)==0){
65             send_state("off");
66         } else{
67             send_state("on");
68         }
69     }
70 }
```

APÊNDICE B - CÓDIGO TELEGRAM

Listing 2. bot.py

```
from telegram.ext import Updater,
    CommandHandler
```

```

2 from telegram import KeyboardButton,
  ReplyKeyboardMarkup, ReplyKeyboardRemove
3 from telegram.ext.filters import Filters
4 from emoji import emoji

6
7 import logging
8 import json
9 import serial
10
11 import serial
12
13 configs = {}
14
15 with open('.conf', 'r') as f:
16     configs = json.loads(f.read())

17 logging.basicConfig(level=logging.DEBUG,
18                     format='%(asctime)s - %(name)s - %(
19                         levelname)s - %(message)s')

20 def send_message(msg):
21     ser = serial.Serial(configs['port']) # open
22     serial port
23     ser.write(str.encode(msg))
24     ser.close()

25 def read_return_message(msg):
26     ser = serial.Serial(configs['port']) # open
27     serial port
28     ser.write(str.encode(msg))
29     ser.flush()
30     ret = ser.readline()
31     ser.close()
32     return ret.decode()

33
34 def turn_on(bot, update):
35     send_message('n')
36     bot.send_message(text=emoji(':bulb:',
37                             use_aliases=True), chat_id=update.message.
38                             chat_id)

39 def turn_off(bot, update):
40     send_message('f')
41     bot.send_message(text=emoji(':new_moon:',
42                             use_aliases=True), chat_id=update.message.
43                             chat_id)

44 def check_state(bot, update):
45     state = read_return_message('s')
46     bot.send_message(text='Lights are {}'.format
47                     (state), chat_id=update.message.chat_id)

48
49 updater = Updater(token=configs['token'])
50
51 # updater.bot.send_message
52 updater.dispatcher.add_handler(CommandHandler(
53     'on', callback=turn_on))
54 updater.dispatcher.add_handler(CommandHandler(
55     'off', callback=turn_off))
56 updater.dispatcher.add_handler(CommandHandler(
57     'state', callback=check_state))

```

```

updater.start_polling()
56 updater.idle()

```

REFERÊNCIAS

- [1] F. Moraes, A. Amory, N. Calazans, E. Bezerra, and J. Petrini, "Using the can protocol and reconfigurable computing technology for web-based smart house automation," in *Integrated Circuits and Systems Design, 2001, 14th Symposium on*. IEEE, 2001, pp. 38–43.
- [2] A. GhaffarianHoseini, N. D. Dahlan, U. Berardi, A. GhaffarianHoseini, and N. Makaremi, "The essence of future smart houses: From embedding ict to adapting to sustainability principles," *Renewable and Sustainable Energy Reviews*, vol. 24, pp. 593 – 607, 2013. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1364032113001342>
- [3] S. Ovadia, "Automate the internet with "if this then that"(ifttt)," *Behavioral & social sciences librarian*, vol. 33, no. 4, pp. 208–211, 2014.