# PONTO DE CONTROLE 2 - FECHADURA ELETRÔNICA CONTROLADA POR BOT DE TELEGRAM

*Jessica Souza, Daniel Auler*

Programa de Graduação em Engenharia Eletrônica, Faculdade Gama
Universidade de Brasília
Gama, DF, Brasil
email: jkoliveiras@outlook.com, danielauler7@gmail.com

## 1. FUNCIONAMENTO

O funcionamento do sistema se dá pela conexão entre o Raspberry e o MSP430 que controla um servo motor para trancar e destrancar portas.
A utilização do raspberry se justifica pelo fato do MSP430 necessitar de módulos externos para se conectar a internet, o que elevaria não só o custo final do projeto como também a complexidade. Como se trata de um projeto viável MVP (produto mínimo viável), a simplificação de tarefas é primordial para garantir a entrega no final do semestre.

### 1.1. Raspberry

O Raspberry utilizado foi um Raspberry Pi Zero W que já conta com módulo de Wifi e por ser pequeno, permite ser acomodado junto ao MSP430 na colocação do projeto. Foi criado um script em python 2.7 utilizando da biblioteca telepot para efetuar a conexão á API do telegram e permitir a comunicação entre o bot e o Raspberry. Segue o código utilizado:

```
// tele_micro.py
import time, datetime
import RPi.GPIO as GPIO
import telepot
from telepot.loop import MessageLoop

led = 26
now = datetime.datetime.now().ctime()
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
 #LED
GPIO.setup(led, GPIO.OUT)
GPIO.output(led, 0) #Off initially

def action(msg):
   chat_id = msg['chat']['id']
   command = msg['text']
   print 'Received: %s' % command
   if 'open' in command:
      message = "Unlocked "
      if 'door' in command:
         message = message + "door" + " at
             " + now
         GPIO.output(led, 1)
         telegram_bot.sendMessage (chat_id,
             message)

   if 'close' in command:
      message = "Locked "
      if 'door' in command:
         message = message + "door " + "at
             " + now
         GPIO.output(led, 0)
         telegram_bot.sendMessage (chat_id,
             message)

telegram_bot =
    telepot.Bot('792286575:AAFZp72JW32M5o7vOafWiOnqqj
print (telegram_bot.getMe())

MessageLoop(telegram_bot,
    action).run_as_thread()
print 'Up and Running....'

while 1:
   time.sleep(10)
```

Tratamos a comunicação como um led, quando desejamos destrancar a porta, mandamos um sinal HIGH no bit do let e o contrário para trancar a porta. Utilizamos como comando do bot as palavras open e close, para evitar um comando não intencional, adicionamos a validação de ter a palavra door na mensagem.

### 1.2. MSP430

O MSP430 foi programado utilizando o software Energia com bibliotecas prontas. Utilizamos a biblioteca ¡servo.h¿

para facilitar o controle do servo motor que tranca e destranca porta. Segue o código do MSP430 no Energia:

```
#include <Servo.h>

Servo myservo; // create servo object to
    control a servo
            // a maximum of eight servo
                objects can be created
const int pinServo = 2;
const int pinEntrada = 3;

int pos = 0; // variable to store the servo
    position
int doorState = 0; //0 = locked, 1 =
    unlocked
int changed = 0;

void unlock() {
  for(pos = 0; pos < 90; pos += 1) // goes
      from 0 degrees to 180 degrees
  {                          // in steps of
      1 degree
    myservo.write(pos);      // tell servo
        to go to position in variable 'pos'
    delay(15);               // waits 15ms
        for the servo to reach the position
  }
}

void lock() {
  for(pos = 90; pos>=1; pos-=1) // goes
      from 180 degrees to 0 degrees
  {
    myservo.write(pos);      // tell servo
        to go to position in variable 'pos'
    delay(15);               // waits 15ms
        for the servo to reach the position
  }
}

void setup()
{
  myservo.attach(pinServo); // attaches the
      servo on pin 9 to the servo object
  pinMode(pinEntrada, INPUT);
}


void loop()
{
  doorState = digitalRead(pinEntrada);
  if (changed != doorState) {
    if (doorState == HIGH) {
      unlock();
      changed = doorState;
    }
    else {
      lock();
      changed = doorState;
    }
  }
}
```

Utilizamos duas funções responsáveis por trancar e destrancar e dentro da main gerenciamos qual função deve ser chamada.

Para conectar o Raspberry no MSP430 utilizamos apenas um jumper, recebendo o sinal led do Raspberry como pinEntrada no MSP430.

## 2. PRÓXIMOS PASSOS

Os próximos passos do projeto consistem em criar uma segurança na comunicação do bot para impedir que qualquer pessoa consiga controlar a porta de todos os usuários. Para isso utilizaremos um banco não relacional mongoDB onde salvará as informações do usuário e do equipamento e sempre irá verificar se quem está enviando os comandos tem as permissões para controlar uma porta específica.
Outro passo importantíssimo é a transferência do código do Energia para o Code Composer convertendo o uso das bibliotecas em funções otimizadas para o sistema requerido.