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

Daniel Auler

Programa de Graduação em Engenharia Eletrônica, Faculdade Gama
Universidade de Brasília
Gama, DF, Brasil
email: 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

Não houve alteração no raspberry mas 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 Code composer. Utilizamos um gerenciamento do clock interno de

1MHz sabendo que para a correta utilização do servo deveríamos utilizar 1ms para rodar para a direita e 2ms de pulso para rodar para a esquerda. Assim criamos um código que recebe o valor do Rasp e executa uma função que roda o motor devidamente para o lado solicitado. Segue o código inicial:

```
#include <msp430.h>

#define SERVO BIT6
#define PERIODO 20000
#define MIN_T 500 //1000
#define MAX_T 2500 //2000
#define STEP_T 50
#define TELEGRAM_DATA BIT4 // Port 1.3

int main(void)
{
   WDTCTL = WDTPW + WDTHOLD;
   BCSCTL1 = CALBC1_1MHZ;
   DCOCTL = CALDCO_1MHZ; //Clock
       configuration
   P1DIR |= SERVO;
   P1SEL |= SERVO;
   P1DIR &= ~BIT3 ; //explicitly making
       P1.3 as Input - even though by
       default its Input
   P1REN = BIT3; //Enable Pullup/down
   P1OUT = BIT3; //Select Pullup

   TACCR0 = PERIODO-1;
   TACCR1 = MIN_T;
   TACCTL1 = OUTMOD_7; //TimerA as RESET/SET
   TACTL = TASSEL_2 + MC_1;
   _BIS_SR(CPUOFF); //Turnoff CPU
   int value = P1IN & TELEGRAM_DATA;
   while(1)
   {
      if (!(P1IN & BIT3) ) {
         TACCR1 += STEP_T;
               if(TACCR1>=MAX_T)
               {
                  TACCR1 = MAX_T;
                  value = 0;
               }
      }
      else
      {
         TACCR1 -= STEP_T;
               if(TACCR1<=MIN_T)
               {
                  TACCR1 = MIN_T;
                  value = 0;
               }
      }
   }
}
```

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

2. PRÓXIMOS PASSOS

Um dos próximos passos do projeto consiste em construírmos um sistema com baixo consumo de energia, seria interessante implementar um código com interrupção para que o MSP430 só fosse ativado quando realmente fosse necessário. Outro ponto que ficou pendente está 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.