

# Sistema de Rastreamento Solar

Eficiência na geração de energia limpa

Julia Pessoa Souza  
Universidade de Brasília - UNB  
Brasília-DF, Brasil  
juliapessoasouza@gmail.com

Victor Barreto Batalha  
Universidade de Brasília - UNB  
Brasília-DF, Brasil  
victor.batalha@hotmail.com

**Resumo**— A busca por eficiência e a melhor obtenção de energia tem se tornado uma constante no estudo da geração de energia solar e com o uso da MSP430 pode-se fazer um modelo de rastreamento de luz para placas solares.

**Palavras chaves**— energia; eficiência; MSP430;

## I. JUSTIFICATIVA

Nos tempos atuais, as discussões no viés de energia renovável, em virtude do aumento do aquecimento global do planeta, têm sido crescentes. O sol é considerado uma fonte de energia sustentável e inesgotável do ponto de vista humano. Tendo em vista a necessidade do maior aproveitamento desta fonte de energia uma solução é o rastreamento de luz para placas solares. Esta é uma forma eficiente de aumentar o aproveitamento da energia [5].

O sistema proposto otimiza a captação de energia solar por meio do rastreamento de luz. Assim, a mesma placa fotovoltaica pode gerar mais eletricidade ocupando a mesma área, o que aumenta o aproveitamento da energia e a eficiência da geração de eletricidade por meio desta fonte. O sistema diminui o ângulo de incidência entre a luz e o painel solar, aumentando a porcentagem da produção de energia daquele painel [1]. Isso diminui a possível perda de aproveitamento da luz por conta da mudança de posição do sol ao longo do dia[2].

Os materiais necessários para se construir o sistema são 2 sensores LDR, 2 resistores de 10k Ohms, 1 servo-motore, jumpers, uma protoboard para a montagem do circuito auxiliar e da placa MSP430 para configurar a lógica programacional.

O sistema de seguimento de luz solar consegue aumentar em até 50% a captação de luz no verão e 20% no inverno [6]. Além de ser necessário menos espaço para gerar a mesma quantidade de energia, um sistema de rastreamento solar também é capaz de entregar a potência de forma mais uniforme, ou seja, há uma máxima produção de energia por

mais tempo ao longo do dia. Conclui-se que o sistema tem capacidade para aproveitar melhor a captação desta fonte de energia[3].

## II. DESENVOLVIMENTO

### A. Hardware

O hardware do projeto descrito foi representado em um diagrama de blocos para facilitar a observação.

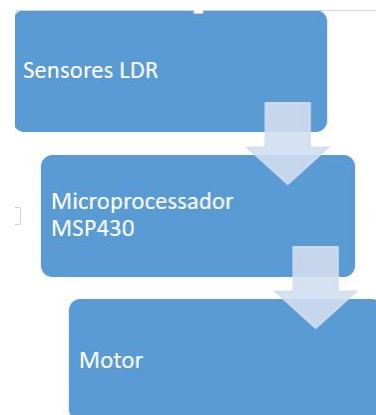


Figura.1.- Diagrama de Blocos

### 1.) Sensor LDR

A identificação do surgimento e da localização da luz será feita pelo sistema através do sensor chamado LDR, acoplado o mesmo junto de um resistor no circuito final.



Figura.2-Sensor LDR

O LDR significa resistor dependente de luz ou seja, quanto maior a incidência de luz menor a resistência do mesmo [7]. “Tal componente é constituído de um semiconductor de alta resistência, que ao receber uma grande quantidade de fótons oriundos da luz incidente, ele absorve elétrons que melhoram sua condutibilidade, reduzindo assim sua resistência.

## 2.) MSP 430

Como em nossa disciplina trabalhamos com o MSP430, trabalharemos como sugerido pelo professor a versão MSP-EXP430G2. Toda a programação lógica para a resolução do sistema e do problema proposto será embarcada e coloca nessa parte do sistema, sendo responsável por parte da simulação em código via o software Code Composer Visual da texas instruments [4].



Figura.3- MSP430

## 3.) Servo Motor

Escolhemos o motor DC devido a facilidade em diversas situações como por exemplo poder operar em constante reversão, operar em corrente contínua, sua velocidade ser ajustável e ao seu alto torque na partida podendo assim se movimentar a placa mais rapidamente [8].



Figura.4 - Servo Motor

Através da ideia e realizamos a montagem de um esquemático do circuito proposto no proteus para melhor visualização do sistema sendo representado na figura a seguir.

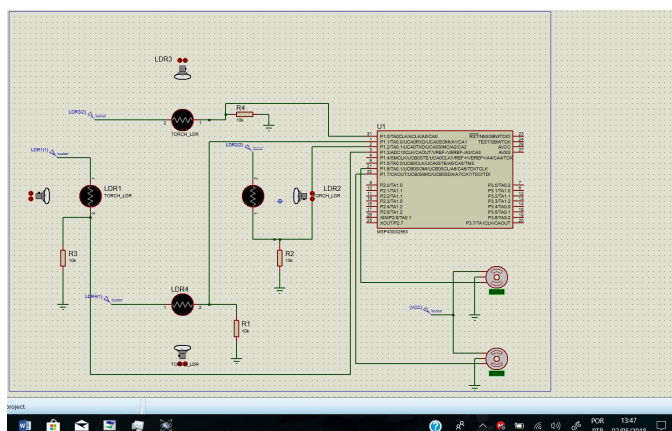


Figura.5 - Esquemático do Circuito

## B. Software

O código do projeto foi feito usando linguagem C, foi testado no sistema Energia, que converte de um código Arduino para MSP430. O esquemático do circuito foi feito no programa Proteus para a realização de testes. O código foi feito definindo limites para os dois motores, para limitar a placa solar de realizar uma rotação completa. Foi utilizada uma média dos valores dos sensores superiores, dos inferiores, dos da parte esquerda e dos da parte direita. Assim a rotação de cada servo motor pode ser melhor definida. Se a média dos valores superiores estiver maior do que a dos inferiores, o motor que está posicionado na horizontal, gira um pouco mais para o lado superior, e, no caso contrário, gira mais para o lado inferior. O motor que está posicionado na vertical gira mais para o lado direito se a média do lado direito estiver maior, e mais para o esquerdo no caso contrário. Assim o motor vai girando até chegar ao seu limite definido. O limite é testado no circuito até atingir um valor razoável para a rotação da placa. O código foi compilado com sucesso no software energia. Na figura pode-se observar a quantidade de memória utilizada.

```

solartrackArduino | Energia 1.6.10E18
Arquivo  Editor  Sketch  Ferramentas  Ajuda

solartrackArduino
{
  pinMode(LED_BUILTIN, OUTPUT);
}

vertical.write(servo);
}

if (abs(tol > dhoriz) || dhoriz > tol) // check if the difference is in the tolerance else change horizontal angle
{
  if (av1 > av2)
  {
    servo = --servo;
    if (servo < servohLimitLow)
    {
      servo = servohLimitLow;
    }
  }
  else if (av1 < av2)
  {
    servo = ++servo;
    if (servo > servohLimitHigh)
    {
      servo = servohLimitHigh;
    }
  }
  horizontal.write(servo);
}
delay(dtime);
}

Compilação terminada

O sketch usa 1.556 bytes (3%) de espaço de armazenamento para programas. O máximo são 131.072 bytes.
Variáveis globais usam 288 bytes (3%) de memória dinâmica, deixando 7.904 bytes para variáveis locais. O máximo são 8.192 bytes.

```

Figura.6 - Código no software Energia

### III. RESULTADOS

### IV. REFERENCIA

- [1] <http://www.byd.com/br/pv/sts.html>
- [2] Projeto de um sistema de rastreamento solar baseado na teoria de controle por servovisão - CEFET/RJ
- [3] Estudo comparativo entre metodos de rastreamento solar aplicados a sistemas fotovoltaicos
- [4] Manual do MSP430 disponivel em:<<https://github.com/Victor-Barreto-Batalha/Microcontroladores-1/tree/master/Refs/MSP430>>
- [5] “Energia solar”, um breve resumo”, Aneel. Acesso em 04/09/2017. Disponivel em: <[http://www2.aneel.gov.br/aplicacoes/atlas/pdf/03-energia\\_solar\(3\).pdf](http://www2.aneel.gov.br/aplicacoes/atlas/pdf/03-energia_solar(3).pdf)>
- [6] “Em que consiste um sistema seguidor solar fotovoltaico”, Portal Energia, Acesso em 02/04/2018. Disponivel em <<https://www.portal-energia.com/em-que-consiste-sistema-seguidor-solar-fotovoltaico/>>
- [7] Material sobre o LDR disponivel em:<<https://portal.vidadesilicio.com.br/sensor-de-luz-com-ldr/>>
- [8] Material sobre o Motor DC disponivel em: <<http://www.kalatec.com.br/o-que-sao-motores-dc/>>

## Apêndice

```
#include <Servo.h>
// 180 maximo horizontal
Servo horizontal; // servo horizontal
int servoh = 180; // 90; // padrao servo horizontal
int servohLimitHigh = 180;
int servohLimitLow = 65;
// limites do servo horizontal
Servo vertical; // servo vertical;
int servov = 45; // 90; // padrao servo vertical
int servovLimitHigh = 80;
int servovLimitLow = 15;
//limites servo horizontal
// LDR pin connections
// name = analogpin;
int ldrlt = 0; //LDR top left - sensor superior esquerdo
int ldrrt = 1; //LDR top right - sensor superior direito
int ldrl = 2; //LDR down left - sensor inferior esquerdo
int ldrr = 3; //ldr down right - sensor inferior direito
void setup()
{ Serial.begin(9600);
// pinos dos servos no arduino
horizontal.attach(9);
vertical.attach(10);
//posicao padrao dos motores
horizontal.write(180);
vertical.write(45);
delay(3000);
}
void loop()
{ int lt = analogRead(ldrlt);
int rt = analogRead(ldrrt);
int ld = analogRead(ldrl);
int rd = analogRead(ldrr);
// tolerancia dos motores
int dtime = 10; int tol = 50;
int avt = (lt + rt) / 2; // average value top - valor entre os superiores
int avd = (ld + rd) / 2; // average value down - valor entre os inferiores
int avl = (lt + ld) / 2; // average value left - valor entre os sensores da parte esquerda
int avr = (rt + rd) / 2; // average value right - valor entre os sensores da parte direita
int dvert = avt - avd; // diferenca entre os sensores superiores e inferiores (motor vertical)
int dhoriz = avl - avr; // diferenca entre os sensores da parte direita e esquerda (motor horizontal)
```

```

//checar se a diferenca esta dentro da tolerancia, se nao estiver, mudar o angulo vertical (comentario)
if (-1*tol > dvert || dvert > tol)
{
if (avt > avd) // se houver mais luz na parte superior, girar placa verticalmente mais para a parte superior, so ate seu limite
{
servov = ++servov;
if (servov > servovLimitHigh)
{
servov = servovLimitHigh;
}
}
// Se nao, girar verticalmente para a parte inferior, ate seu limite.
else if (avt < avd)
{
servov = --servov;
if (servov < servovLimitLow)
{
servov = servovLimitLow;
}
}
vertical.write(servov);
}
if (-1*tol > dhoriz || dhoriz > tol) // check if the diffrence is in the tolerance else change horizontal angle
{
if (avl > avr)
{
servoh = --servoh;
if (servoh < servohLimitLow)
{
servoh = servohLimitLow;
}
}
else if (avl < avr)
{
servoh = ++servoh;
if (servoh > servohLimitHigh)
{
servoh = servohLimitHigh;
}
}
horizontal.write(servoh);
}
delay(dtime);
}

```

