Universidade de Brasília Faculdade UnB Gama

CIRCUITO CARREGADOR DE ENERGIA SOLAR

Prática de Circuitos Eletrônicos 2 - 119458
Turma B

Nome:	Matrícula:	Local:	Data:
Ítalo Rodrigo Moreira Borges	15/0012446	Gama -DF	07/07/2017
Leonardo Amorim de Araújo	15/0039921	Gama -DF	07/07/2017

Sumário

1. Introdução

A sustentabilidade atualmente é um dos grandes objetivos da humanidade, buscar novas formas de obter energia mais limpa é uma tendência para ter um futuro sem muitos transtornos, e a energia eólica e a energia solar estão em alta como uma das principais energias para substituir as não renováveis.Visando a necessidade futura,construiu-se um circuito que carrega baterias na faixa de 1 volt até 6.8 volts. O resultado final com vídeo pode ser visto na referência [8].

2. Objetivos

Construção de um carregador de bateria utilizando luz solar com faixa de tensão ajustável. Este tem a possibilidade de girar a base da placa solar conforme a luz do sol varia durante o dia.

3. Parte Pré-Experimental

3.1. Esquemático do Circuito do Carregador de Baterias

Na figura 1 é mostrado o esquemático do circuito utilizado para realizar a carga de baterias. O circuito é utilizado para carregar baterias de chumbo ácido ou de níquel-cádmio e é projetado para carregar baterias de 6V e até 4.5 Ah para diversas aplicações. Este ainda possui regulagem de tensão e corrente e atua desacoplando a bateria quando esta está completamente carregada para que não enxergue o circuito carregado como uma carga.

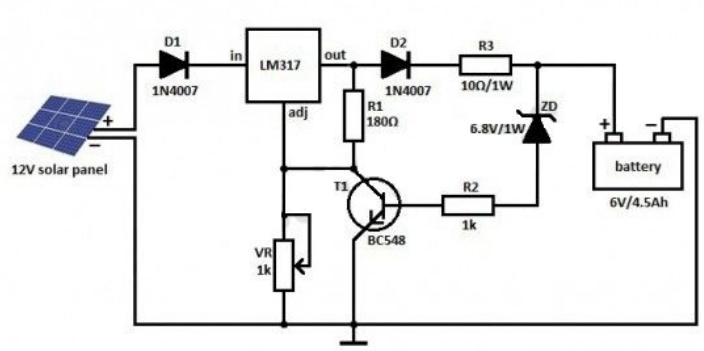


Figura 1 - Esquemático do circuito que realiza a carga das baterias. Fonte [1]

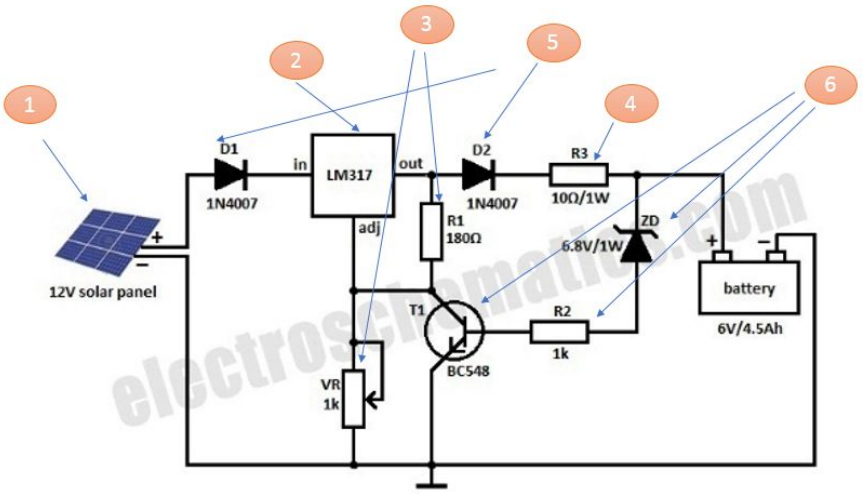


Figura 2 - Análise da função de cada componente no circuito

Na figura 2, o componente indicado com **1** é placa solar utilizada para fornecer energia ao circuito. O componente **2** é o CI LM317, um regulador de tensão variável. Conforme a referência [6], o LM317 é capaz de fornecer até 1.5 A de corrente, com uma faixa de trabalho para tensão de entrada entre 1.5 V até 37 V. Sua vista de frente pode ser visualizada na figura 3.

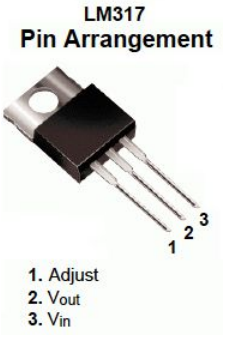


Figura 3 - Arranjo de pinos do CI LM317

O circuito interno do LM317 pode ser visto na figura 4. Pode-se visualizar que este possui um amplificador interno, um diodo zener que é acionado com 1.25 V, dois TBJs NPN atuando como par Darlington e um circuito de proteção contra alta temperatura e altas correntes (o fabricante não deixa claro qual é este circuito na referência [6]).

Os resistores R1 e VR em **3** realizam o ajuste de tensão no circuito. A fórmula básica do ajuste é dada por

$$V_{out} = 1.25V\,(1 + (R2/R1)) + (I_{adj} \times R2) \quad (1)$$

onde Iadj é a corrente que retorna do pino de ajuste de tensão.

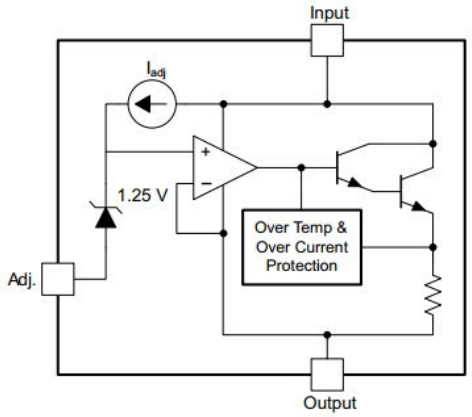


Figura 4 - Bloco Funcional do LM317 mostrando seus componentes internos

O diodo em **5** tem a função de previne a descarga de corrente da bateria para o circuito. Logo, quando a tensão da bateria atinge seu máximo, ainda que esta tenha uma tensão maior que a tensão do carregador, a bateria é desacoplada do circuito por este diodo.

Os componentes em **6** atuam quando a bateria está carregada. Como o diodo D2 impede que a bateria enxergue o circuito como carga, este faz com que o diodo zener de Vz = 6.8 V conduza e acione a base do transistor. Como a base é acionada, o carregador de bateria é habilitado para transmitir uma corrente pelo coletor, sendo o resistor de 180 Ω o resistor Rc, ou resistor de carga do coletor.

Quando o carregador está atuando para carregar uma bateria, os componentes em 6 não atuam, devido a não polarização reversa do diodo zener.

3.2. Esquemático do circuito que alimenta o microcontrolador, sensores e servomotores

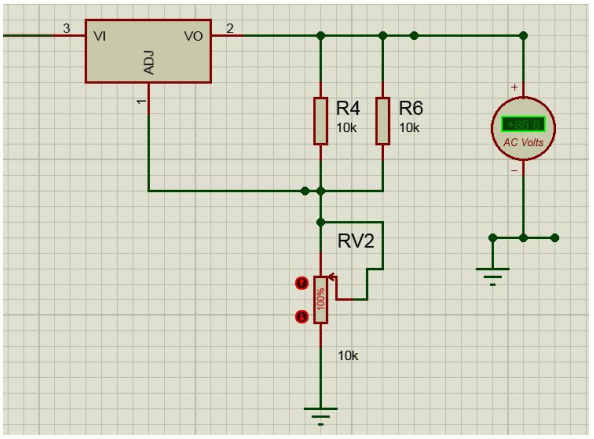


Figura 5 - Circuito para realizar a alimentação do microcontrolador e seus módulos

O circuito na figura 5 tem a mesma lógica do circuito da figura 2, mas este só deve realizar regulação de tensão. A faixa de trabalho devido a fórmula (1) é de 1.25 V até 1.25 x (1 + 10k/5k) = 3.75 V. Somando-se a corrente de ajuste, teremos então uma tensão máxima na

saída Vo de 4,25 V. O microcontrolador utilizado neste circuito opera com tensão de 3 V, logo a regulação realizada por este circuito se enquadra perfeitamente na faixa de operação deste. A tensão de entrada é a tensão fornecida pela placa solar de 12V.

3.3. Esquemático do circuito que controla o painel solar

Na figura 6 encontra-se o esquemático utilizado para controle da base da placa solar. O objetivo é que conforme a Terra movimenta-se durante o dia, a placa solar possa estar posicionada de forma que favoreça o melhor posicionamento para absorção da luz solar.

3.3.1. Light Dependent Resistor - LDR

Os LDRs utilizados no circuito são os sensores para verificação do estado atual da luz incidindo sobre a placa. O LDR é um resistor cuja resistência varia conforme a intensidade de luz. Especificamente, se a intensidade de luz aumenta, sua resistência diminui. O LDR é constituído de um material semicondutor com elevada resistência elétrica. A figura 7 mostra o componente.

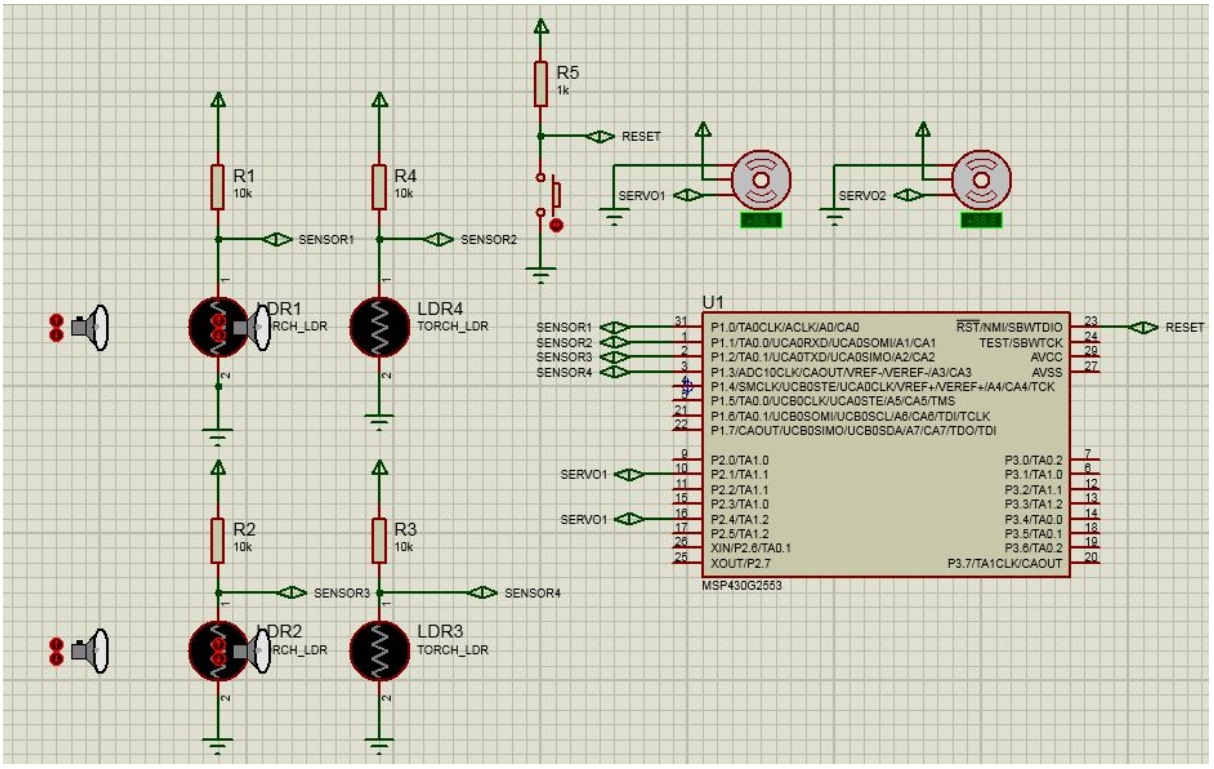


Figura 6 - Esquemático mostrando o MCU MSP430g2553 e os servos motores, assim como os LDRs e os resistores limitadores de corrente. O push button utilizado é devido ao pino RST da MSP430, que deve permanecer em nível alto para funcionamento normal e em nível baixo por um instante de tempo para reiniciar o microcontrolador.

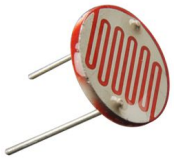


Figura 7 - Típico LDR

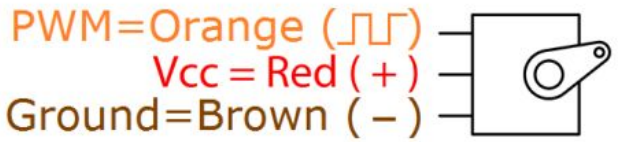
Devido ao divisor de tensão formado pelos resistor de 10kΩ e o LDR, que fica variando muito conforme a incidência de luz, nas portas P1.0 até a P1.3 do microcontrolador foi habilitado o uso do conversor A/D interno ao MSP. O conversor A/D é chamado de ADC10 e implementa uma conversão de 10 bits. Portanto, têm-se 1024 valores de tensão possíveis, variando de 0 V até 3 V com um degrau de 3/1024 ≅ 3 mV de tensão para cada dígito digital.

3.3.2. Servomotor

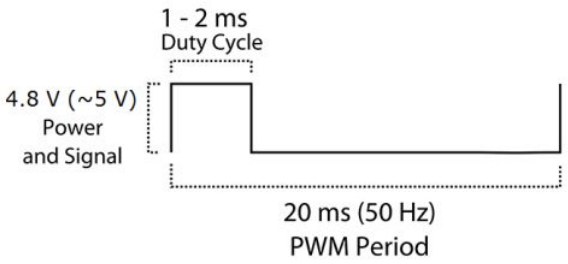
Um servomotor é um tipo de máquina eletromecânica que pode ser controlada por comandos elétricos. O servomotor, diferente dos motores elétricos convencionais, possui somente a liberdade de girar o eixo 180º (em alguns modelos 360º), tendo uma grande precisão quanto à angulação. O servomotor possui três componentes básicos em seu interior:

- O sistema atuador que possui o motor elétrico, a maioria um motor elétrico de corrente contínua, como o utilizado neste projeto;
- Um sensor que em grande parte dos servos comercializados é um potenciômetro que é acoplado ao eixo do motor que indica a posição angular atual do servo;
- Um circuito eletrônico que recebe o sinal do sensor (vindo do eixo) e um sinal externo de controle que serve para acionar o motor para a posição desejada.

O modelo de servomotor utilizado neste projeto é o SG90, que possui um torque de 1.8 kgf.cm, velocidade de operação de 0.1s/60º, tensão de operação de 3~5 V. A figura 8 mostra como é o funcionamento deste componente.



(a)



(b)

Figura 8 - (a) Forma de ligação do pinos do servomotor (b) Modo de Operação do Servomotor e forma de controle do eixo utilizando PWM, conforme a referência [9].

Pode-se visualizar na figura 8 que o servomotor tem 3 pinos, sendo que um deles é o chamado PWM - Pulse Width Modulation, que em tradução livre, significa Modulação por Largura de Pulso. No datasheet, conforme a referência [9], indica que a forma de onda deve ter frequência de 50 hz, se repetindo portanto a cada 20 ms. O ciclo de trabalho (Duty Cicle) na Figura 8(b) indica quanto tempo a forma de onda deve permanecer no estado ‘1’, ou ‘nível lógico alto’. Esse período em que a forma de onda permanece em nível lógico alto é que irá definir qual será a angulação do servomotor. Para o caso do SG90, tem-se que 0 ms equivale à -90º e 2 ms equivale a 90º, supondo que a forma de onda começa em t = 0. A figura 9 mostra o modelo de servomotor utilizado.

Figura 9 - Modelo de SG90 Utilizado

3.3.3. Microcontrolador MSP430 modelo g2553

O microcontrolador MSP430 vem de uma família de microcontroladores RISC de 16 bits voltados para aplicações de baixo a baixíssimo consumo, fabricado pela Texas Instruments. O MSP430 modelo g2553 possui:

- Frequência de Operação de até 16 Mhz;
- Memória RAM de 500 KB;
- Pinos de GPIO: 24 (depende do modelo);
- Analog to Digital Converter 10 bits - ADC10: Possui 8 canais;
- 2 Timers com modo de comparação e captura;
- Tensão Máxima de Operação: 3.6 V.

Este microcontrolador foi escolhido para o projeto de propósito, pois para projetos envolvendo trabalhos com energia solar, consumo de energia é fator crítico. Para se ter uma ideia, no modo de baixo consumo máximo, o MSP consome em torno de 0.1 uA. Logo isso o torna o melhor microcontrolador existe hoje no mercado para aplicações de baixo consumo.

3.3.4. Código Utilizado

No Apêndice A encontra-se o código utilizado para o controle de posição dos eixos da base da placa solar. A linguagem utilizada é C. Iremos realizar aqui somente uma explicação geral do objetivo de cada função utilizada no código. Mais detalhes sobre o microcontrolador MSP430 e seus modos de operação podem ser consultados na referência [10].

- Setup_ADC: Realiza a configuração inicial dos pinos P1.0 até P1.3 para serem utilizados como canais do conversor A/D. São nestes pinos que os sensores são conectados para tratamento dos dados.

- **Setup_Servos**: Define as portas P2.1 e P2.4 como saídas do Timer1, onde OUT1 e OUT2, que são saídas do canal de comparação do Timer1, estarão definidas nestes dois pinos. Este dois pinos que são utilizados para o controle de posição dos servomotores.

- **Setup_LED**: Função de teste utilizada para verificar o funcionamento do código. Esta serve para setar os LEDS da Launchpad para verificar o pleno funcionamento do código.

- **Read_ADC**: Função que lê os valores convertidos em cada canal do conversor A/D ADC10 e armazena os valores na memória, onde estes valores estarão disponíveis no vetor de inteiros adc[], sendo que

adc[0] - Corresponde ao Sensor R0
adc[1] - Corresponde ao Sensor R1
adc[2] - Corresponde ao Sensor L1
adc[3] - Corresponde ao Sensor L2

A figura 10 mostra como são definidos esses sensores e como é o giro de cada servomotor.

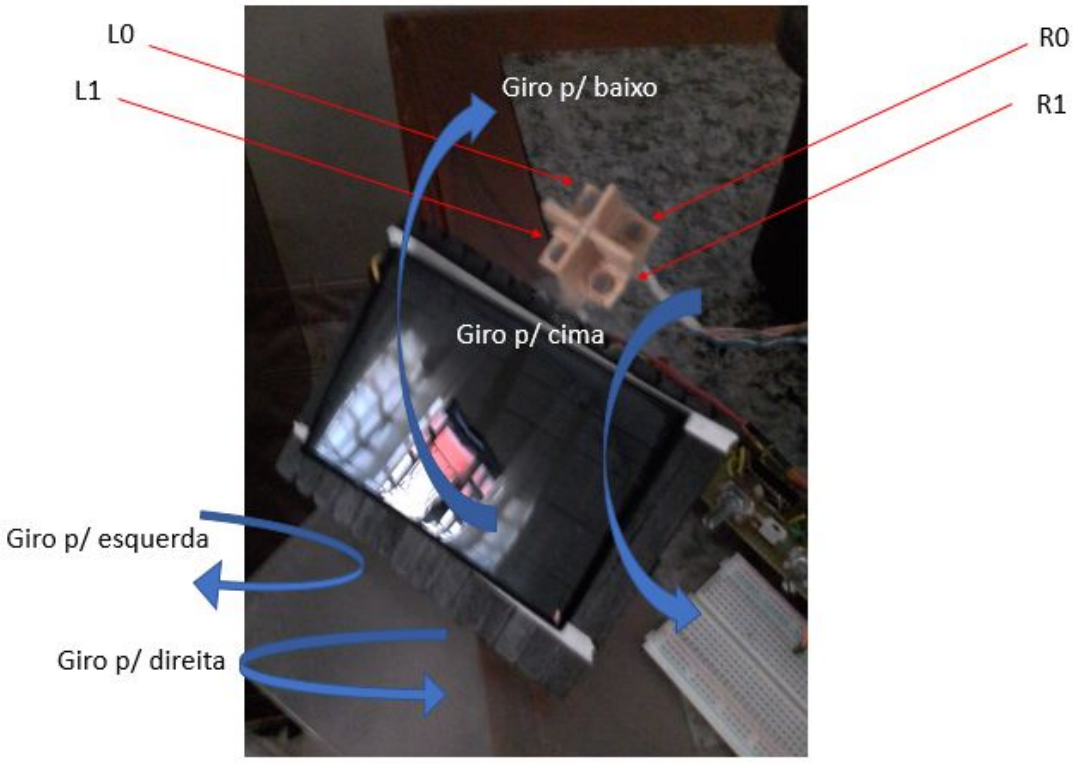


Figura 10 - Ilustração de programação dos giros do servomotor e a ordem dos sensores

É nesta função ainda que podemos ver a seguinte linha de comando:

__bis_SR_register(CPUOFF + GIE);

Este comando habilita a interrupção da MSP e desliga o clock principal, reduzindo bastante o consumo de energia. Quando o conversor terminar de realizar a conversão com o clock secundário, ele habilita a interrupção do conversor, que pode ser visto na função de interrupção

```
// ADC10 interrupt service routine
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
  __bic_SR_register_on_exit(CPUOFF);        // Clear CPUOFF bit from 0(SR)
}
```

Que aciona o clock novamente para o prosseguimento do código.

- **Read_LDRS:** Esta função trabalha com os valores lidos nos sensores L0, L1, R0 e R1 e age atuando sobre os servomotores para que estes se movam ou não conforme um feedback anterior. Este feedback anterior é a variável média, que recebe os valores das médias dos sensores que têm valores próximos na última vez que a função foi chamada.

Quando existe maior incidência de luz sobre um sensor (ou sensores), estes ficam com a tensão mais baixa do que os outros, e dessa forma é possível identificar qual é a angulação de incidência de luz. Por exemplo, se nos sensores L0 e R0 houve maior incidência de luz, eles terão um valor de tensão lido pelo conversor menor do que a média geral de tensão lido em todos os sensores e dessa forma, controla-se os servomotores para que girem para baixo, ou seja, inclinem-se de forma a ficar na horizontal com o servo de elevação.

Mas se por exemplo, somente o servo R0 tiver maior incidência de luz em seu lado, deve-se girar para a direita o servomotor que controla o eixo azimutal.

Esta função portanto é a mais importante do código, pois faz o papel de ligação entre os valores lidos pelos sensores e o acionamento dos servomotores conforme estes valores.

Servos_PWM: Esta função é que trata os dados recebidos pela função Read_LDRs, aumentando ou diminuindo o PWM de cada servomotor conforme a inclinação que deve ser feita para pegar a melhor incidência de luz.

4. Parte Experimental

4.1. Materiais Utilizados

- 2 Diodo 1N4007
- 2 LM317
- 1 Resistor 10 Ω / 1 W ± 5%
- 1 Resistor 180 Ω / [1 ÷ 4 W] ± 5%
- 2 Resistor 1k Ω / [1 ÷ 4 W] ± 5%
- 6 Resistor 10k± 5%
- 1 Potenciômetro [1 K Ω] ± 5%
- 1 Potenciômetro [10 k Ω]± 5%
- 1 Transistor BC548 [NPN]
- 1 Diodo Zener 1N4736 (6,8V/1W)
- 1 Célula Painel Placa Energia Solar Fotovoltaica 12V/1.5 W
- 1 Suporte Pan-Tilt com 2 servomotores incluídos
- 1 Microcontrolador MSP430g2553
- 4 Resistores Dependentes de Luz - LDRs
- 1 Push-Button

4.2. Resultados

Na figura 4 mostra o circuito carregador de energia solar construído na protoboard a fim de testar e verificar o funcionamento do circuito. Utilizou-se multímetro digital para verificar a tensão de saída e a corrente a fim de avaliar e explorar a capacidade do circuito.

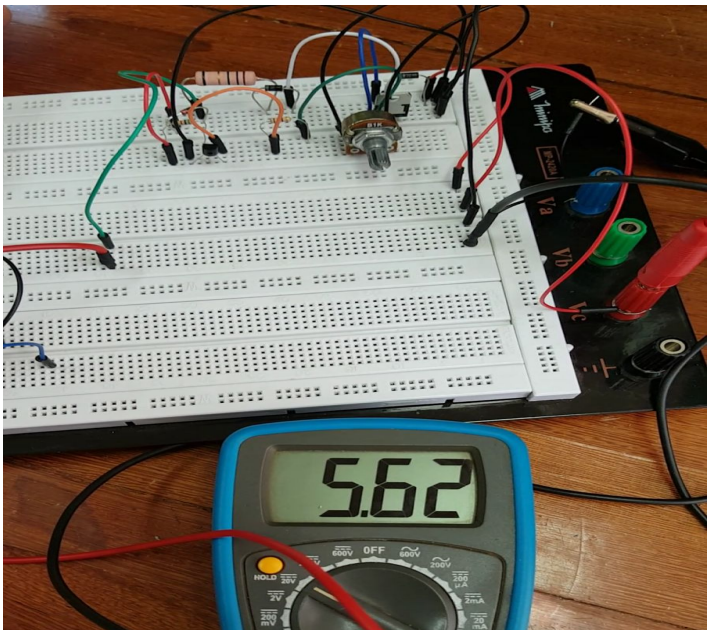


Figura 11 - Circuito montado em protoboard

A figura 12 mostra o circuito carregador de energia solar com a placa fotovoltaica e uma bateria de 3.6 volts, com o auxílio do multímetro observou-se as pequenas variações de tensão e corrente, que não afetaram a carga da bateria. Além disso, pode-se verificar que a tensão que a placa fotovoltaica forneceu ao circuito foi maior que a especificação (12 Volts).

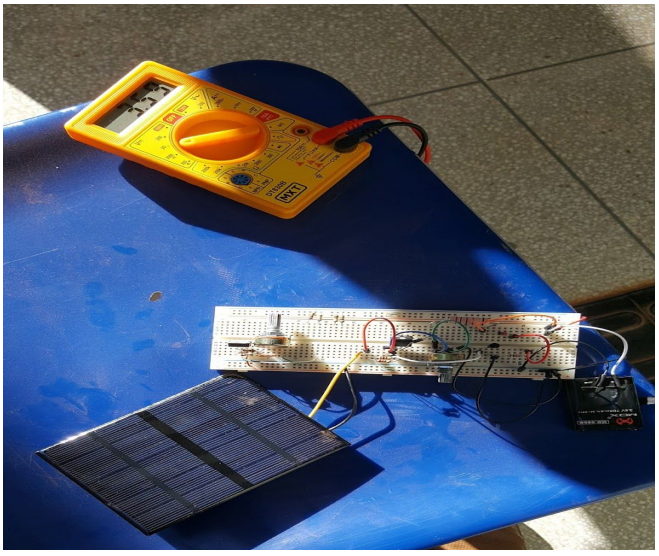


Figura 12 - Circuito montado em protoboard com a Placa solar e a Bateria de 3.6 V

Na figura 13 mostra o circuito carregador de energia solar implementado em uma placa de circuito,onde posicionou-se cada componente eletrônico e soldou-se cada nó do circuito.

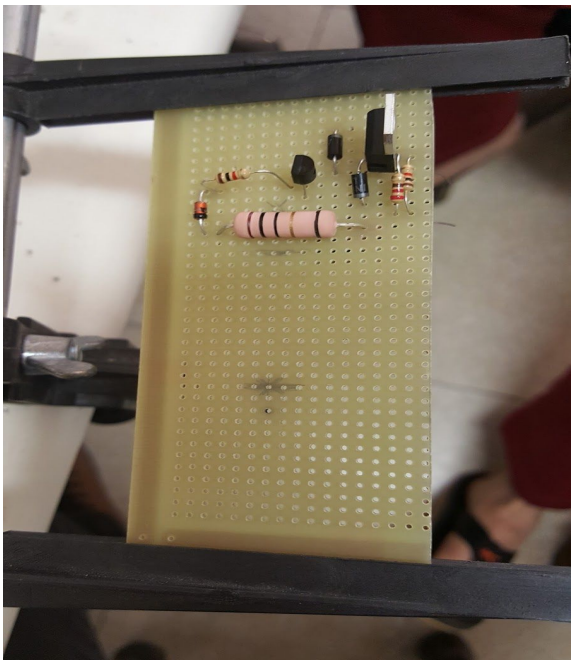


Figura 13 - Circuito sendo soldado

O circuito montado completamente, com os componentes do carregador solar e do regulador para o microcontrolador pode ser visualizado na Figura 14.

Figura 14 - Circuito completo montado e soldado na placa

4.2.1. Teste em um celular

Utilizamos o carregador de baterias e ligamos este a um celular do modelo Samsung Galaxy Pocket. A tensão de saída da placa solar regulada que já começou a carregar a bateria do celular foi de 4.42 V, mostrando a eficiência do projeto.

Figura 15 - Teste do Carregador de Baterias em um celular - Tensão de Saída

A figura 16 mostra o circuito do carregador de baterias carregando um celular do modelo Samsung Galaxy J1.

Figura 16 - Circuito Carregador de Baterias carregando um celular na luz do dia

Um vídeo completo mostrando os resultados pode ser consultado na referência [8].

5. Discussão

5.1. Pontos positivos

- Fornece energia;
- Tem a função de carregar diversos dispositivos;
- O Circuito tem um custo baixo;
- Produto para concorrer no mercado;
- Sustentável;
- O Brasil possui uma das melhores condições para gerar este tipo de energia.

5.2. Pontos negativos

- A Placa fotovoltaica tem um custo elevado;
- O tempo para carregar uma bateria ainda é demorado.

5.3. Custo do projeto

Componentes do Circuito do Carregador Solar

Componente	Preço	Quantidade	Total	Link
Diodo 1N4007	R$ 0,07	2x	R$ 0,14	https://goo.gl/BLbQIE
LM317	R$ 1,06	2x	R$ 2,12	https://goo.gl/Fqmmnc
Resistor 10R 5W	R$ 0,52	1x	R$ 0,52	https://goo.gl/iFsWM8
Transistor BC548 NPN	R$ 0,16	1x	R$ 0,16	https://goo.gl/IA3fnh
Diodo Zener 1N4736 (6,8V/1W)	R$ 0,19	1x	R$ 0,19	https://goo.gl/MA6u7n
Potenciômetro de 1K	R$ 1,10	1x	R$ 1,10	https://goo.gl/4gED7a
Potenciômetro de 10K	R$1,10	1x	R$ 1,10	https://goo.gl/T5w3uY
Resistor de 1K	R$ 0,04	2x	R$ 0,08	https://goo.gl/F

				Hf4XF
Resistor de 10K	R$ 0,04	2x	R$ 0,08	https://goo.gl/y5JXLE
TOTAL			R$ 4,97	

Componentes da Base do Carregador Solar

Componente	Preço	Quantidade	Total	Link
Suporte Pan-Tilt com 2 servomotores incluídos	R$ 58,90	1x	R$ 58,90	https://goo.gl/X0euEB
TOTAL			R$ 58,90	

Comparação do Circuito construído x Circuito de Mercado

Produto	Preço	Link
Carregador Original 20000mah Pineng Pn-969 Bateria Externa	R$ 110,00	https://goo.gl/c9nmrn
Carregador Solar Portátil	R$ 60,97	https://goo.gl/1Uscrt

5.2. Observações

Nos primeiros testes verificou-se um superaquecimento do potenciômetro, mas que foi corrigido quando ajustamos a faixa de operação do mesmo, logo, houve uma corrente regular que passava pelo potenciômetro. Nos dias em diante não houve problema, teve-se alguns testes para avaliar o comportamento do circuito e depois de avaliado, concluiu-se o projeto, soldando cada componente do circuito em uma placa.

6. Conclusão

O funcionamento do circuito carregador de energia solar requer a incidência da radiação solar para excitar as células solares constituídas de materiais semicondutores, isso

gera uma diferença de potencial ocasionando a passagem de lacunas e elétrons. Após essa fase primordial, alimentará o circuito carregador de bateria solar. O circuito foi dividido em 4 estágios, o primeiro é **ajuste de tensão e corrente** que é executado pelo LM317, potenciômetro e o resistor de $180\,\Omega$, o segundo é a **restrição do carregamento** que é executada pelo resistor de $10\,\Omega$ para não mandar uma corrente muito alta para a carga, o terceiro estágio é a **interrupção do corte do carregamento** que é executada pelo transistor BC548 NPN e o quarto estágio é **impedimento da descarga da bateria após a mesma está carregada** que é executada pelos diodos, onde o diodo 1N4007 tem a função de não deixar carga da bateria para o circuito.

7. Referências Bibliográficas

[1] Circuito de Carregador Solar de Baterias
Fonte: http://blog.novaeletronica.com.br/circuito-de-carregador-solar-de-baterias/

[2] Solar Charger for 6V Battery,
Fonte: http://www.electroschematics.com/4746/solar-charger-circuit/

[3] MERCADO DE ENERGIA SOLAR NO BRASIL,
Fonte: http://www.portalsolar.com.br/mercado-de-energia-solar-no-brasil.html

[4] Camera Based Solar Tracking System
Fonte: https://www.youtube.com/watch?v=ZS_8bXmswb4&t=148

[5] Dual Axis "Smart" Solar Tracker
Fonte: https://browndoggadgets.com/products/dual-axis-smart-solar-tracker

[6] Datasheet: LM317 3-Terminal Adjustable Regulator, Texas Instruments
Fonte: http://www.ti.com/lit/ds/symlink/lm317.pdf

[7] Datasheet: Mixed Signal Microcontroller - MSP430, Texas Instruments
Fonte: http://www.ti.com/lit/ds/symlink/msp430g2253.pdf

[8] Vídeo do carregado realizado pela dupla funcionando 100%:
Fonte: https://www.youtube.com/watch?v=kGw1LaIk0_M

[9] Datasheet do Servomotor SG90
Fonte: http://www.micropik.com/PDF/SG90Servo.pdf

[10] Davies, John H. MSP430 Microcontroller Basics. 2008. 30 Corporate Drive, Suite 400, Burlington, MA01803, USA. Elsevier Ltd.

8. Anexos

A. Códigos

```
#include <msp430g2553.h>
#define SERVO1 BIT1 //P2.1 as TA1.1
#define SERVO2 BIT4 //P2.4 as TA1.2
#define LED BIT6
#define ADC_CHANNELS 4
#define PARAMETRO_MEDIA 10
#define GIRO_DIREITO 100
#define GIRO_ESQUERDO -100
#define GIRO_CIMA 100
#define GIRO_BAIXO -100
#define ATRASO_GIRO 25
#define L0 adc[3]
#define L1 adc[2]
#define R1 adc[1]
#define R0 adc[0]
unsigned int adc[ADC_CHANNELS];
volatile unsigned int media=0;
unsigned int comparacao;
unsigned int init=0;
unsigned int new_config=0;
unsigned int P_TA1CCR1=0;
unsigned int P_TA1CCR2=0;

void Atraso(volatile unsigned int x)
{
   TA0CCR0 = 1000-1;
   TA0CTL |= TACLR; //clear timer
   TA0CTL = TASSEL_2 + ID_0 + MC_1;
   while(x>0)
   {
     x--;
     while((TA0CTL&TAIFG)==0);
           TA0CTL &= ~TAIFG;
   }
   TA0CTL = MC_0; //stop timer
}

void Setup_LED()
{
  P1OUT &= ~BIT6; //Inicializar LED desligado
```

```
  P1DIR |= BIT6;
}

void Acender_LED(char string[])
{
  if(string == "ON")
  {
    P1OUT |= LED;
  }
  else if(string == "OFF")
  {
    P1OUT &= ~LED;
  }
  else if (string == "PISCAR")
  {
    P1OUT ^= LED;
    Atraso(500);
  }
}

void Setup_Servos(void)
{
  P2DIR |= SERVO1+SERVO2;
  P2SEL |= SERVO1+SERVO2;
  P2SEL2 &= ~(SERVO1+SERVO2);
}

void Servos_PWM(int azimute, int elevation)
{
  if(init==0)
  {
    TA1CTL |= TACLR;
    TA1CCR0 = 20000;
    TA1CCR1 = 1600;
    TA1CCR2 = 1300;
    TA1CCTL1 = OUTMOD_7;
    TA1CCTL2 = OUTMOD_7;
    TA1CTL = TASSEL_2 + ID_0 + MC_1;
    init++;
    P_TA1CCR1 = TA1CCR1;
    P_TA1CCR2 = TA1CCR2;
  }
//criterio para eixo polar
  if((TA1CCR1 > 600) && (TA1CCR1 < 2600))
  {
    P_TA1CCR1 = TA1CCR1+azimute;
```

```
    new_config=1;
  }
  else if((TA1CCR1==600) && (azimute > 0))
  {
    P_TA1CCR1 = TA1CCR1+azimute;
    new_config=1;
  }
  else if((TA1CCR1==2600) && (azimute < 0))
  {
    P_TA1CCR1 = TA1CCR1+azimute;
    new_config=1;
  }
//criterio para elevacao
  if((TA1CCR2>600)&&(TA1CCR2<1300))
  {
    P_TA1CCR2 = TA1CCR2+elevation;
    new_config=1;
  }
  else if((TA1CCR2==600) && (elevation > 0))
  {
    P_TA1CCR2 = TA1CCR2+elevation;
    new_config=1;
  }
  else if((TA1CCR2==1300) && (elevation < 0))
  {
    P_TA1CCR2 = TA1CCR2+elevation;
    new_config=1;
  }
//criterio para nova configuracao
  if(new_config==1)
  {
    TA1CTL |= TACLR;
    TA1CCR0 = 20000;
    TA1CCR1 = P_TA1CCR1;
    TA1CCR2 = P_TA1CCR2;
    TA1CCTL1 = OUTMOD_7;
    TA1CCTL2 = OUTMOD_7;
    TA1CTL = TASSEL_2 + ID_0 + MC_1;
    new_config=0;
  }
}

void Setup_ADC(void)
{
  ADC10CTL0 |=  SREF_0 +  ADC10SHT_0 + MSC + ADC10ON + ADC10IE; //Reference
from
```

```
  //Vcc and Vss, sampling time of 16×ADC10CLKs, ADC10ON
  ADC10CTL1 |= INCH_3 + CONSEQ_1 + ADC10SSEL_3 + SHS_0; //Input channel A3, A2,
A1 and A0; repeated sequence
  ADC10AE0 |= (BIT0+BIT1+BIT2+BIT3); // Analog Input in P1.0, P1.1, P1.2 and P1.3;
  ADC10DTC1 = ADC_CHANNELS; // 4 conversions
  ADC10CTL0 |= ENC + ADC10SC; // Sampling and conversion start
}

void Read_LDRS()
{
  //L0 - adc[3] - Sensor da Esquerda 0
  //L1 - adc[2] - Sensor da Esquerda 1
  //R0 - adc[0] - Sensor da Direita 0
  //R1 - adc[1] - Sensor da Direita 1
  if(media!=0)
  {
    comparacao = media - PARAMETRO_MEDIA;
     if((L0 < comparacao) && (R0 < comparacao))
     {
       //os dois sensores de cima detectaram maior intensidade de luz
       //girar para cima
       Servos_PWM(0,GIRO_BAIXO);
       media=(R0+L1)/2;
     }
     else if ((L1 < comparacao) && (R1 < comparacao))
     {
      //os dois sensores de baixo detectaram maior intensidade de luz
      //girar para baixo
       Servos_PWM(0,GIRO_CIMA);
       media = (R1 + L0)/2;
     }
     else if (L0 < comparacao)
     {
       //sensor da esquerda 0 detectou maior intensidade de luz
       //girar para a esquerda
       Servos_PWM(GIRO_ESQUERDO,0);
       media = (R0+R1+L1)/3;
       Atraso(ATRASO_GIRO);
     }
     else if (L1 < comparacao)
     {
       //sensor da esquerda 1 detectou maior intensidade de luz
       //girar para a esquerda
       //Rotate_Servos(3);
       Servos_PWM(GIRO_ESQUERDO,0);
       media = (L0+R0+R1)/3;
```

```
      Atraso(ATRASO_GIRO);
    }

    else if (R1 < comparacao)
    {
      //sensor da direita 0 detectou maior intensidade de luz
      //girar para a direita
      //Rotate_Servos(4);
      Servos_PWM(GIRO_DIREITO,0);
      media = (L0+L1+R0)/3;
      Atraso(ATRASO_GIRO);
    }
    else if (R0 < comparacao)
    {
      //sensor da direita 1 detectou maior intensidade de luz
      //girar para a direita
      //Rotate_Servos(4);
      Servos_PWM(GIRO_DIREITO,0);
      media = (L0+L1+R1)/3;
      Atraso(ATRASO_GIRO);
    }
    else
    {
    media = (adc[0] + adc[1] + adc[2] + adc[3])/4;
    }
  }
  else
  {
     media = (adc[0] + adc[1] + adc[2] + adc[3])/4;
  }
}

void Read_ADC(void)
{
  ADC10CTL0 &= ~ENC;
  while (ADC10CTL1 & BUSY);// Wait if ADC10 core is active
  ADC10SA = (unsigned int)adc;// Copies data in ADC10SA to unsigned int adc array
  ADC10CTL0 |= ENC + ADC10SC;
 __bis_SR_register(CPUOFF + GIE);
  Read_LDRS();
}

int main( void )
{
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;
```

```
  BCSCTL1 = CALBC1_1MHZ;
  DCOCTL = CALDCO_1MHZ;
  Setup_ADC();
  Setup_Servos();
  Servos_PWM(0,0);
  Atraso(1000);
  Setup_LED();
  for(;;)
  {
    Read_ADC();
  }

  return 0;
}

// ADC10 interrupt service routine
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
  __bic_SR_register_on_exit(CPUOFF);        // Clear CPUOFF bit from 0(SR)
}
```