UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETROTÉCNICA
CURSO DE ENGENHARIA INDUSTRIAL ELÉTRICA

CLEVERSON AUGUSTO DA SILVA SANTOS
HEROS AUGUSTO ANTUNES SILVA
JARDEL LEONARDI DE CARVALHO

# DESENVOLVIMENTO DE UM MECANISMO DE POSICIONAMENTO DE PLACAS FOTOVOLTAICAS PARA RASTREAMENTO SOLAR VIA RASPBERRY PI

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2015

CLEVERSON AUGUSTO DA SILVA SANTOS
HEROS AUGUSTO ANTUNES SILVA
JARDEL LEONARDI DE CARVALHO

# DESENVOLVIMENTO DE UM MECANISMO DE POSICIONAMENTO DE PLACAS FOTOVOLTAICAS PARA RASTREAMENTO SOLAR VIA RASPBERRY PI

Trabalho de Conclusão de Curso de Graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do curso de Engenharia Industrial Elétrica com ênfase em Automação, do Departamento Acadêmico de Eletrotécnica - DAELT - da Universidade Tecnológica Federal do Paraná – UTFPR, como requesito parcial para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Amauri Amorin Assef.

CURITIBA
2015

**Cleverson Augusto da Silva Santos**
**Heros Augusto Antunes Silva**
**Jardel Leonardi Carvalho**

# Desenvolvimento de um mecanismo de posicionamento de placas fotovoltaicas para rastreamento solar via Raspberry PI

Este Trabalho de Conclusão de Curso de Graduação foi julgado e aprovado como requisito parcial para a obtenção do Título de Engenheiro Eletricista, do curso de Engenharia Industrial Elétrica ênfase Automação do Departamento Acadêmico de Eletrotécnica (DAELT) da Universidade Tecnológica Federal do Paraná (UTFPR).

Curitiba, 02 de julho de 2015.

_____
Prof. Paulo Sérgio Walenia, Esp.
Coordenador de Curso
Engenharia Industrial Elétrica ênfase Automação

_____
Profa. Annemarlen Gehrke Castagna, Mestre
Responsável pelos Trabalhos de Conclusão de Curso
de Engenharia Industrial Elétrica ênfase Automação do DAELT

**ORIENTAÇÃO**

_____
Amauri Amorin Assef, Dr.
Universidade Tecnológica Federal do Paraná
Orientador

**BANCA EXAMINADORA**

_____
Amauri Amorin Assef, Dr.
Universidade Tecnológica Federal do Paraná

_____
Roberto Cesar Betini, Dr.
Universidade Tecnológica Federal do Paraná

_____
Ohara Kerusauskas Rayel, Dr.
Universidade Tecnológica Federal do Paraná

## AGRADECIMENTOS

Gostaríamos de agradecer aos nossos pais que sempre nos apoiaram em todas as fases de nossas vidas, aos professores que nos ensinaram tudo que sabemos sobre engenharia elétrica e acima de tudo a Deus que nos guiou até aqui e continua a nos mostrar o caminho a ser seguido.

Em especial ao pai do Jardel, Sr. Jandir José de Carvalho, que nos ajudou com soluções para a construção da estrutura do projeto e ao Prof. Dr. Amauri Amorin Assef que nos auxiliou disponibilizando seu tempo, conhecimento e contribuição com a placa “MICROPIC_DAELT Versão 0” que permitiu uma solução mais simples para o projeto.

Por fim, o nosso muito obrigado a todos que contribuíram direta ou indiretamente para que esse trabalho fosse realizado, os nossos sinceros agradecimentos.

## RESUMO

SANTOS, S. A. C.; SILVA, A. A. H.; CARVALHO, L. J.; **Desenvolvimento de um mecanismo de posicionamento de placas fotovoltaicas para rastreamento solar via Raspberry PI**. 2015. 108f. Trabalho de conclusão de curso (Graduação em Engenharia Elétrica). Universidade Tecnológica Federal do Paraná, Curitiba, 2015.

O projeto trata do desenvolvimento de um mecanismo de posicionamento com dois graus de liberdade, podendo movimentar uma placa solar nos eixos polares, através de dois motores DC. O primeiro motor fica acoplado no painel que faz a rotação no primeiro eixo, vertical. O segundo motor fica na base da estrutura que faz a rotação da estrutura no segundo eixo polar, horizontal. São usados quatro sensores fotoresistores para identificar para onde a estrutura deve se mover de modo a permanecer no ponto de maior incidência de luz solar. O controle dos motores é feito via Raspberry PI e um microcontrolador. O projeto visa a utilização do mesmo para determinação da viabilidade econômica da implementação de painéis com controle em detrimento do uso de painéis fixos nos mesmos locais, tanto quanto a utilização do mesmo para o ensino na universidade.



**Palavras-chave**: Raspberry PI. Rastreamento Solar. Placas Fotovoltaicas.

## ABSTRACT

SANTOS, S. A. C.; SILVA, A. A. H.; CARVALHO, L. J.; **Desenvolvimento de um mecanismo de posicionamento de placas fotovoltaicas para rastreamento solar via Raspberry PI**. 2015. 108f. Trabalho de conclusão de curso (Graduação em Engenharia Elétrica). Universidade Tecnológica Federal do Paraná, Curitiba, 2015.

The Project is about the development of a positioning mechanism with two degrees of freedom, which is able to move a solar panel on polar axes, through two DC motors. The first motor is attached to the panel allowing the rotation on the first axis, vertical. The second motor stays on the structure's base which allows the rotation on the second polar axis, horizontal. There are four photosensors used to identify where the structure should move to, in a way that allows it to remain on the spot with the most sunlight incidence. The motors control is made by Raspberry PI and a microcontroller. The Project aims to the resolution of the economic viability of the panel's implementation using controlled motors over the use of fixed panels on a same location, as well on the university classes.



**Key words**: Raspberry PI. Tracking. Photovoltaic Panel.

## LISTA DE TABELAS

## LISTA DE FIGURAS

## LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

| | |
|---|---|
| A4 | Consumidores de modalidades tarifárias, cobrados tanto pela demanda quando pelo consumo |
| ARM | Advanced Risk Machine |
| CRESESB | Centro de Referência para Energia Solar e Eólica |
| DB | *Database* |
| GPIO | *General Purpose Input Output* |
| HDMI | *High Definition Multimedia Interface* |
| HTTP | *HyperText Transfer Protocol* |
| I2C | *Inter Integrated Circuit* |
| JSON | *JavaScript Object Notation* |
| LAN | *Local Area Network* |
| LDR | *Light Dependant Resistor* (Resistor Dependente de Luz) |
| OS | *Operational System* |
| PCH | Pequenas Centrais Hidroelétricas |
| PHP | *Hypertext Preprocessor* |
| PN | Junção de dois cristais semicondutores, dopados com elementos que agregam características P ou N de acordo com sua estrutura atômica |
| PWM | *Pulse Width Modulation* |
| Python | Linguagem de programação de Alto Nível |
| PVC | *Polyvinyl Chloride* |
| RC Servo | *Radio Control Servo* |
| SD | *Secure Digital (Memory)* |
| SQL | *Structured Query Language* |
| SSH | *Secure Shell* |
| STC | *Standard Test Conditions* (Teste de Condições Padrão) |
| USART | *Universal Synchronous Asynchronous Receiver Transmitter* |
| UNSEGED | *United Nations Solar Energy Group* |

**SUMÁRIO**

## 1 INTRODUÇÃO

O impacto crescente da emissão de $CO_2$, proveniente dos combustíveis fósseis e da biomassa nas mudanças climáticas, tem levado as economias desenvolvidas a apoiarem estudos para busca de alternativas energéticas que venham a substituir as energias convencionais (COLLE & PEREIRA, 1998).

As fontes de energia existentes utilizam recursos naturais que eventualmente se esgotarão em um futuro próximo e sua utilização traz problemas ambientais irreversíveis ao planeta. As energias renováveis, proveniente dos ventos, dos rios e do sol, são cada vez mais estudas e estão sendo vistas como o futuro energético do planeta (COLLE & PEREIRA, 1998).

A energia solar, além de ser uma energia limpa e inesgotável, é a fonte com maior potencial de geração no planeta, pois origina as demais fontes renováveis de forma indireta. Através do desenvolvimento da tecnologia, o homem utilizou a energia solar de inúmeras formas, captando-a e armazenando-a. Nos tempos modernos, avanços da tecnologia permitiram a criação de painéis solares para a transformação dessa energia em eletricidade. Entretanto, por normalmente serem instalados de forma fixa, não aproveitavam todo seu potencial, mesmo em lugares com alta incidência de radiação solar.

A tecnologia das placas fotovoltaicas atuais é relativamente cara, não existindo ainda tecnologia eficiente com alto rendimento que maximize sua capacidade. Em uma hora, o sol fornece à atmosfera terrestre o que corresponde ao consumo energético mundial de um ano (LUQUE & HEGEDUS, 2011). Com a utilização de placas fixas, a captação na maior parte do dia não é máxima, devido ao ângulo de incidência dos raios na placa. Para maximizar a incidência de luz, tecnologias modernas utilizam sistemas de direcionamento que procuram manter as placas na angulação correta para uma maior incidência solar.

Até o momento foram desenvolvidos e são utilizados dois métodos básicos para o rastreamento solar. O primeiro método é o rastreamento uniforme, sendo baseado na velocidade constante da rotação da terra. O segundo método é chamado de método da comparação da radiação, tendo como princípio a mudança da intensidade da radiação sobre fotoresistores e a modificação dos sinais elétricos.

## 1.1 DELIMITAÇÃO DO TEMA

Muitas placas solares existentes são instaladas de forma fixa em um determinado ângulo, não aproveitando de forma máxima a energia solar que poderia ser captada, devido ao movimento da Terra.

Devido a diferenças geográficas, é difícil saber ao certo se a instalação de painéis solares que se orientam ao sol é economicamente viável comparado à simples instalação do painel em um determinado ângulo fixo.

Normalmente não é identificada quão vantajosa é a utilização de sistemas de rastreabilidade solar. Somente uma análise de custo poderia demonstrar se existe vantagem do investimento econômico do sistema e se há ganho efetivo do dispositivo na localidade estudada, por isso a utilização de um rastreador portátil se torna útil.

## 1.2 PROBLEMAS E PREMISSAS

A questão da energia renovável proveniente do sol é, sem dúvida, uma das problemáticas mais debatidas, tanto na ciência como na política. Entretanto, para a proposição de soluções inovadoras são necessários estudos para avaliação de painéis solares. Dessa forma, algumas questões podem ser levantadas. As informações provenientes de uma placa ou sistema fotovoltaico (tensão, corrente e potência) podem ser coletadas e gravadas de maneira remota para viabilizar o estudo da implementação de rastreadores solares? Pode ser usado um módulo Raspberry PI em conjunto com placas para a aquisição e controle de motores de posicionamento?

## 1.3 OBJETIVOS

O objetivo deste trabalho é desenvolver uma plataforma de teste capaz de facilitar a análise da eficiência energética de uma placa fotovoltaica, a partir dos dados coletados por um módulo Raspberry PI. Trata-se do desenvolvimento de um sistema de pequena dimensão (tamanho do painel de 52x35 cm), que possibilite habilitar e desabilitar o rastreamento automático de um painel fotovoltaico de acordo com a luz solar, armazenando os dados no módulo utilizado e permitindo sua consulta atráves de qualquer dispositivo conectado à mesma rede sem fio.

### 1.3.1 Objetivos Específicos

- Pesquisar módulos de painel solar de pequeno porte;
- Desenvolver um mecanismo de posicionamento para um rastreamento solar eficiente;
- Possibilitar o posicionamento a partir de 3 modos de operação diferentes, manual (posição fixa), automático por sensores e automático pelo cálculo da posição do sol;
- Adotar um módulo Raspberry PI para controle do sistema de posicionamento e aquisição de dados;
- Possibilitar a troca entre controle automático e manual da posição de conexão remota do Raspberry PI ou através de interface externa via TCP (Ethernet ou Wi-Fi);
- Obter e armazenar em um banco de dados as informações capturadas de tensão, corrente e potência;
- Permitir a extração de dados para análise através de servidor Web (PHP + Banco de dados) de arquivo de texto no formato JSON[1], XML[2] e CSV[3].

## 1.4 JUSTIFICATIVA

Dada a importância dessa fonte renovável, energia solar, o trabalho busca estabelecer fundamentos para analisar a viabilidade da utilização de painéis fotovoltaicos com mecanismo de posicionamento.

O acesso aos dados coletados pelo dispositivo oferece ao usuário final dados para que a viabilidade da implementação de painéis com capacidade de rastreamento seja determinada.

## 1.5 PROCEDIMENTOS METODOLÓGICOS

[1] *JavaScript Object Notation*. Formato leve para intercâmbio de dados computacionais muito usado em aplicativos para Smartphones.
[2] *Extensible Markup Language.* Linguagem de marcação para envio de mensagens, em que os dados são alocados em tags.
[3] Comma Separated Values. Estrutura de formatação de dados em que os valores são separados por vírgulas. Comumente usados em *softwares* como *Microsoft Excel* e *Numbers*.

1. Estudo dos requisitos materiais gerais, mecânicos e elétricos para a construção do rastreador:

A. Avaliação dos materiais necessários para a construção da estrutura de sustentação mecânica, incluindo esquemático e adaptações necessárias;

B. Estudo de sensores e componentes eletrônicos para os conjunto de sensores que identificarão a posição da luz solar;

C. Verificação de componentes para coleta das informações de tensão e corrente provenientes da placa fotovoltaica;

D. Pesquisa sobre motores possíveis para serem utilizados no projeto, incluindo tipo do motor, forma de controle e potência necessária.

2. Desenvolvimento do sistema de aquisição e preparação de dados:

A. Estabelecer conexão física e lógica entre o Raspberry PI e módulos de controle dos motores de posicionamento;

B. Desenvolver o *software* em linguagem de programação Python, para aquisição dos valores provenientes do conversor e adição ao banco de dados;

C. Desenvolver método de pesquisa dos dados através de PHP, utilizando JSON, XML e XLS como formatos de saída.

3. Desenvolvimento da arquitetura de *hardware* e *software* para rastreamento solar:

A. Pesquisar um painel solar para uso em pesquisa, em escala reduzida;

B. Acoplamento mecânico dos dois motores ao painel solar, permitindo seu movimento com 2 graus de liberdade;

C. Desenvolver o *firmware* em linguagem C para movimento dos motores via microcontrolador, e em Python para o sistema de rastreamento no Raspberry Pi.

## 1.6 ESTRUTURA DO TRABALHO

O trabalho será organizado da seguinte maneira:

- Capítulo 1 – Tema, delimitação de tema, problemas e premissas, objetivos gerais, objetivos específicos, justificativa e procedimentos metodológicos;

- Capítulo 2 – Embasamento teórico sobre a situação tecnológica das placas solares, sensores, motores, unidade lógica microcontrolada;
- Capítulo 3 – Desenvolvimento do projeto, considerando os métodos utilizados para efetuar tanto as partes mecânicas (acoplamentos de motores, placa solar e sensores), como todas as ligações elétricas e desenvolvimento de *software;*
- Capítulo 4 – Apresentação dos resultados da plataforma de teste desenvolvida;
- Capítulo 5 – Conclusão.

# 2 FUNDAMENTAÇÃO TEÓRICA

## 2.1 HISTÓRICO DA TECNOLOGIA FOTOVOLTAICA

As primeiras descobertas por trás da relação entre energia luminosa e elétrica foram feitas em 1839 e 1887 pelos cientistas Alexandre E. Becquerel e Heinrich Hertz, respectivamente. Entretanto, o efeito fotoelétrico foi amplamente explicado apenas em 1905, por Albert Einstein. Robert Millikan atestou empiricamente a teoria em 1916. Cerca de 38 anos depois essa tecnologia foi usada para a criação de uma célula capaz de transformar a energia luminosa em energia elétrica (ANDRADE, HOLANDA JR., SYPNIEVSKI, 2006).

Em 1950, uma descoberta possibilitou todo o avanço tecnológico disponível em relação as células fotovoltaicas atualmente. Utilizando a dopagem de semicondutores, os cientistas da época aprenderam a criar semicondutores com junção PN (Figura 1), que é a união de um semicondutor positivo e outro negativo. Quando utilizados dessa forma, os elétrons livres das camadas de valência do semicondutor N ficam livres para transitar de um ponto a outro da junção, transferindo sua carga para a outra região.

Segundo Aldabó (2002), o Efeito Fotovoltaico foi descoberto por Edmund Becquerel, em 1839, mas sua primeira aplicação prática ficou a cargo dos cientistas do Bell Laboratories, na década de 50.

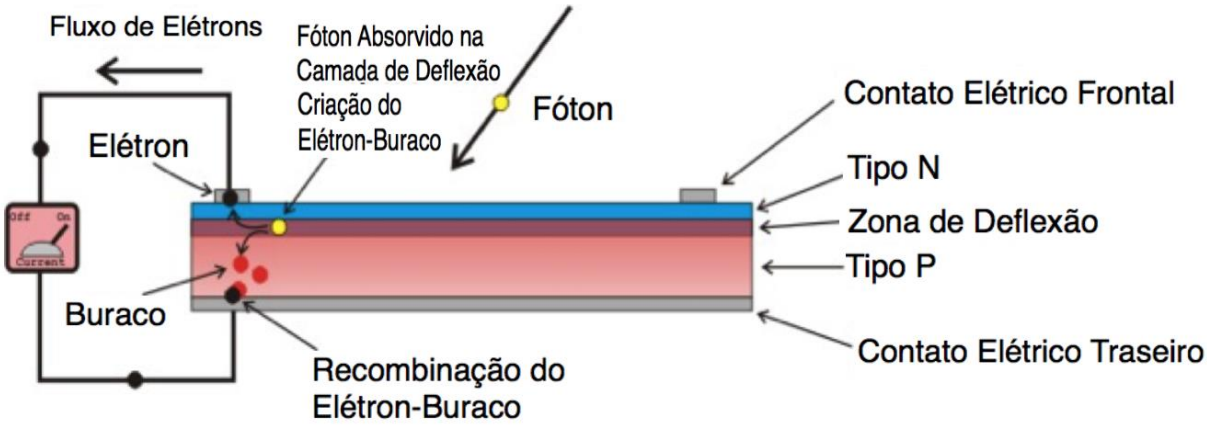



**Figura 1: Estrutura da Junção PN em uma célula fotovoltaica.**

**Fonte: Images Scientific Instruments (2014).**

### 2.1.1 Panorama Energético Mundial

Segundo Andrade (2006 apud COOK, 1995), em meados da década de 70 a crise do petróleo trouxe à tona a necessidade de pesquisa em novas tecnologias de produção de energia elétrica, estando a tecnologia fotovoltaica em destaque pela sua capacidade potencial.

Um dos maiores problemas da tecnologia era o seu custo, que na época chegava a US$ 120,00/Wp[4]. A partir desse momento governo, indústria e a comunidade cientifica começaram um investimento massivo em pesquisa, desenvolvimento e em métodos de produção, o que fez com que em 2001 essa mesma tecnologia custasse US$ 3,50/Wp (ANDRADE, 2006 apud FRAIDENRAICH, 2003). Hoje esse custo pode ser tão baixo quanto US$ 1,00/Wp (ENERGY TREND, 2014).

De acordo com perspectivas da *United Nations Solar Energy Group* (UNSEGED), a participação das energias eólica e solar somadas será superior a 30% da Demanda Global em 2050.

### 2.1.2 Geração Distribuída

A geração distribuída é um conceito de geração que é realizada próxima aos seus consumidores finais. Novas tecnologias permitem acoplamento de diversas fontes de energia como Co-geradores, Geradores de Energia por resíduos combustíveis, Geradores de emergência, Geradores para operação no horário de ponta, painéis fotovoltaicos e PCHs (Pequenas Centrais Hidroelétricas).

Através da utilização de equipamentos de medida, controle e comando é possível controlar os geradores e as cargas para adaptarem-se à oferta de energia. A geração distribuída tem vantagens como a redução de perdas e economia em redes de transmissão (INEE, 2014).

A geração distribuída mostra-se interessante à medida que destaca-se suas vantagens perante outras fontes de energia, como a possibilidade de instalação em

[4] *Watt-peak* - Unidade de potência de pico, linguagem coloquial comercial para a potência nominal da placa fotovoltaica.

grandes telhados de hipermercados, *shoppings centers*, indústria e consumidores A4[5] (ANEEL, 2014).

### 2.1.3 Panorama Geral no Brasil

O Brasil representa um dos países com maior índice de radiação solar, principalmente no Nordeste, onde a insolação solar vai de 4500 kW/m$^2$ até um máximo de 6000 kW/m$^2$. Para efeito de comparação, na Alemanha a maior insolação fica em torno de 3500 kW/m$^2$ (VILLALVA, 2012). Na Figura 2 é apresentado o mapa de irradiação solar no Brasil (SOLARGIS, 2014).

Segundo a Aneel (2014), o Brasil tem um potencial para a utilização de energia solar privilegiado em comparação a outros países na Europa, como ilustrado na Figura 2, onde as cores quentes destacam as regiões com maiores potencais.


**Figura 2: Irradiação Solar no Brasil.**

**Fonte: SolarGIS (2014).**

[5] Consumidores de modalidades tarifárias, cobrados tanto pela demanda quando pelo consumo

O desenvolvimento da tecnologia fotovoltaica no Brasil ainda era pequeno até 2002. Segundo Andrade (2006 apud FRAIDENRAICH, 2003), existiam no Brasil até 2002 mais de 40.000 sistemas instalados, gerando entre 8 e 9 MW de potência. Em 2005, segundo dados da CRESESB, a potência instalada já era de 15 MW, atendendo mais de 180 mil pessoas. Em 2013, a capacidade instalada no País é estimada em 40 MW, sendo que 90% desse total é de projetos com módulos isolados, ou seja, não conectado à rede elétrica (ANEEL, 2014).

## 2.2. CARACTERÍSTICAS DA PLACA FOTOVOLTAICA

### 2.2.1 Silício Policristalino

O Silício Policristalino foi escolhido como material para a placa do projeto pelo seu custo-benefício. Embora sua eficiência energética seja menor que a das células monocromáticas (entre 13% e 15%), a redução do custo de produção faz desta opção uma alternativa interessante (VILLALVA & GAZOLI, 2012).

A Tabela 1 mostra a eficiência energéticas de células fotovoltaicas de diferentes silícios (monocromático, policromático e filme fino), em diferentes meios e montagens (célula em laboratório, célula comercial e módulo comercial).

**Tabela 1: Comparativo de Células Fotovoltaicas x Eficiência.**

| Material da célula fotovoltaica | Eficiência da célula em laboratório | Eficiência da célula comercial | Eficiência dos módulos comerciais |
|---|---|---|---|
| Sílicio Monocristalino | 24,7% | 18,0% | 14,0% |
| Sílicio Policristalino | 19,8% | 15,0% | 13,0% |
| Sílicio Cristalino de Filme Fino | 19,2% | 9,5% | 7,9% |

**Fonte: Villalva e Gazoli (2012).**

### 2.2.2 Módulos Fotovoltaicos

Módulos fotovoltaicos são conjuntos de células fotovoltaicas montadas sobre uma estrutura rígida e conectadas eletricamente em série para que a tensão resultante seja elevada ao patamar desejado.

Os módulos ou painéis encontrados no mercado produzem entre 10 W e 250 W, apresentando uma tensão de até 37 V e corrente de até 8 A. Em contraste com esses valores, as células de filme fino apresentam potências menores de 50 a 100 W, e tensões maiores de até 70 V (VILLALVA & GAZOLI, 2012). Na Figura 3 são apresentadas de forma ilustrativa as principais camadas de materiais que compõem um painel fotovoltaico do tipo Silício Cristalino.

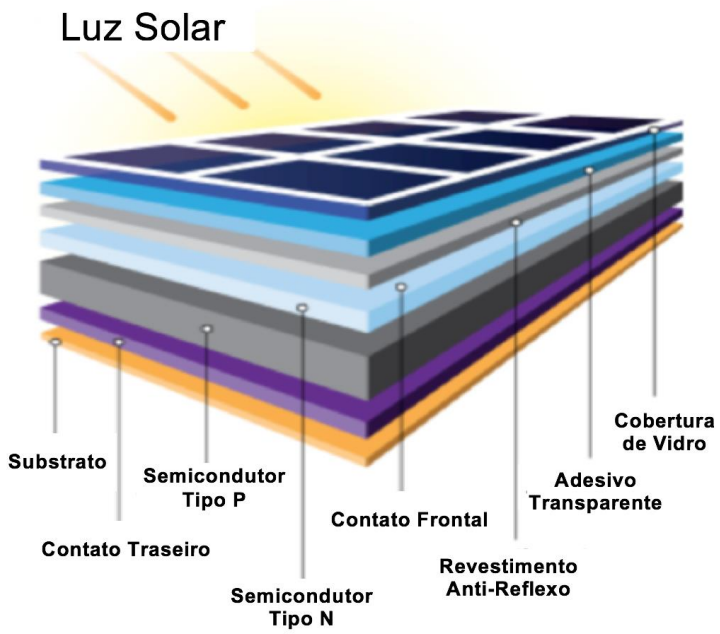



**Figura 3: Componentes de um painel fotovoltaico.**
**Fonte: Portal Ecotech (2014).**

### 2.2.3 Circuito Equivalente

O modelo elétrico, como visto na Figura 4, se inicia com a célula fotovoltaica, que pode ser modelada como uma fonte de corrente contínua. Esta fonte de corrente é afetada por *G* e *Tc*, que representam a energia luminosa incidente e temperatura da célula, respectivamente.

Ao ocorrer uma incidência luminosa capaz de desprender os elétrons da camada de valência ocorre o efeito fotovoltaico, há a criação de pares elétron-lacuna, gerando assim uma corrente proporcional ao nível de radiação (ANDRADE, HOLANDA JR., SYPNIEVSKI, 2006).

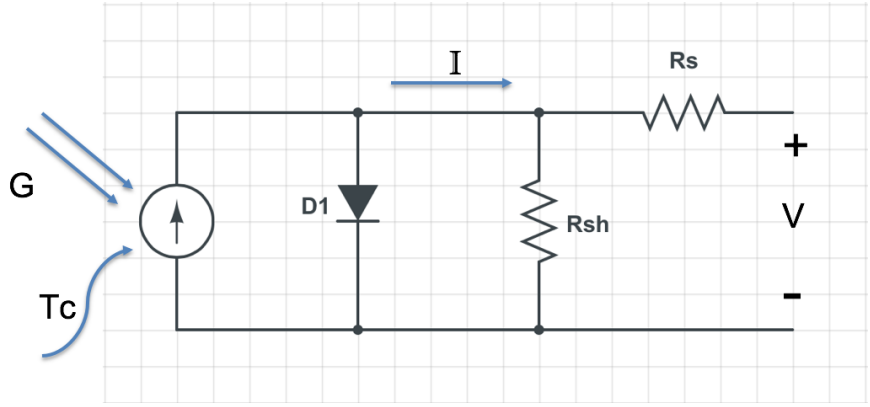



**Figura 4: Circuito Equivalente.**

**Fonte: Holanda Jr. (2006).**

Sendo:

*G*: Energia Luminosa Incidente [W/m$^2$];

*Tc*: Temperatura da célula [ºC];

I: Corrente da saída do painel [A];

V: Tensão de saída do painel [V];

$R_{sh}$: Resistencia Shunt [Ω];

$R_s$:Resistencia Série [Ω];

### 2.2.4 Curvas de corrente, tensão e potência

A tensão da placa fotovoltaica, que não é constante, depende da corrente fornecida. Caso seja conectado aos seus terminais um aparelho que demande muita corrente, a tensão de saída do módulo tende a cair. Do mesmo modo um aparelho que demande pouca corrente fará a tensão do módulo se elevar (VILLALVA & GAZOLI, 2012).

Nas curvas mostradas na Figura 5, o ponto $P_{pm}$ indica a tensão($V_{pm}$) e a corrente($I_{pm}$) para que o módulo forneça a potência máxima. Mostra ainda a correntes de curto circuito($I_{cc}$) e a tensão de circuito aberto($V_{ca}$). Caso a demanda de corrente seja maior ou menor que a do ponto especificado, ocorre uma diminição da potência elétrica, causando uma situação fora da ideal (VILLALVA & GAZOLI, 2012).

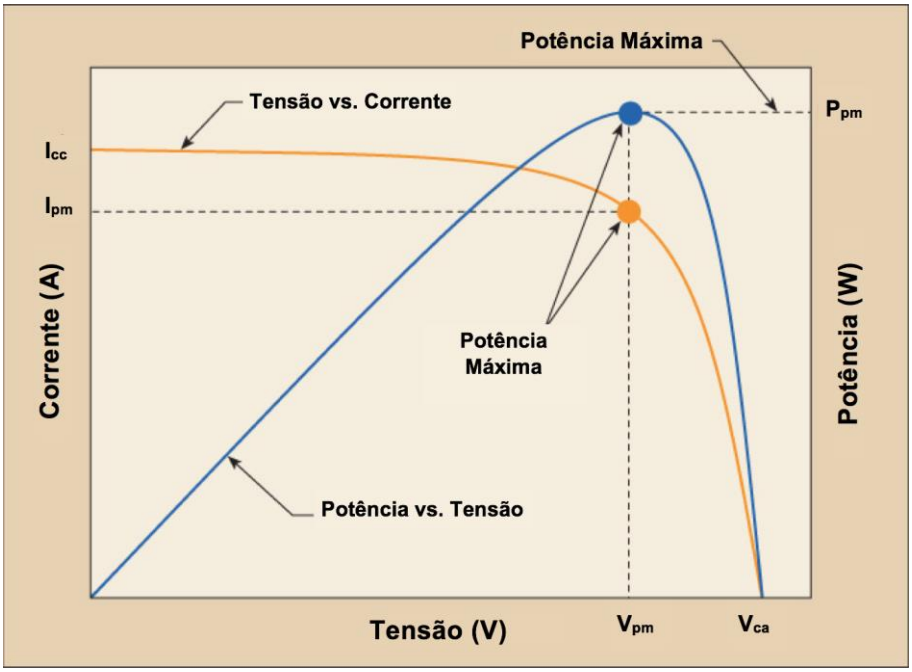



**Figura 5: Corrente x Tensão.**
**Fonte: Electrical Construction & Maintenance (2014).**

### 2.2.5 Influência da Temperatura

Villalva e Gazoli (2012) explicam que a temperatura, quando acima dos limites especificados, influencia negativamente na tensão fornecida pela placa fotovoltaica. Em consequência, existe perda de potência nos módulos, conforme ilustrado na Figura 6.

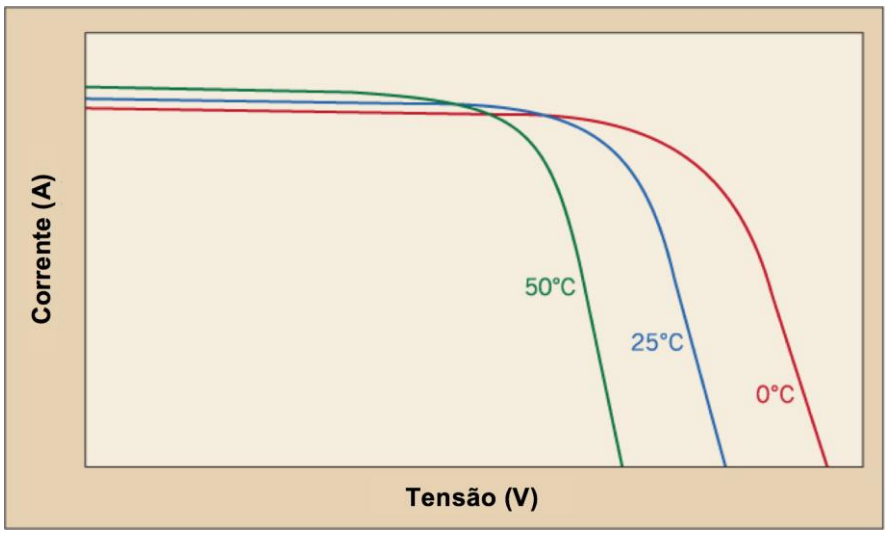



**Figura 6: Influencia da Temperatura na Potência fornecida.**
**Fonte: Electrical Construction & Maintenance (2014).**

### 2.2.6 Informações da Placa Utilizada

A Yingli Green Energy (*www.yinglisolar.com*) é uma das maiores empresas de produção de painéis fotovoltaicos no mundo. A placa utilizada no projeto é uma Yingli YGE 20, construída com células de silício policristalino que tem eficiência energética de 10,9%, eficiência considerada baixa entre as células comerciais de alto desempenho, mas que atende todas as necessidades do projeto, principalmente pelo seu custo-desempenho. A característica de um módulo fotovoltaico comercial é disponibilizada pelos fabricantes em folhas de dados, destacando-se informações como características elétricas e mecânicas.

Nas Tabelas 2 a 5, extraídas do manual do fabricante, pode-se observar informações importantes sobre o painel utilizado, como por exemplo, modelo, potencia, tensão e correntes de operação da placa.

**Tabela 2: Parâmetros da Placa YGE 20 em condição nominal (20°C, 800W/m²).**

| Descrição | Unidade | Valores |
|---|---|---|
| Potência de Saída | W | 14,5 |
| Tensão na Potência Máxima | V | 15,8 |
| Corrente na Potência Máxima | A | 0,92 |
| Tensão do Circuito Aberto | V | 19,4 |
| Corrente de Curto Circuito | A | 1,07 |

**Fonte: Yingli Solar (2014).**

**Tabela 3: Parâmetros da Placa YGE 20 em condição padrão (25 °C, 1000W/m²).**

| Descrição | Unidade | Valores |
|---|---|---|
| Potência de Saída | W | 20 |
| Tolerância da potência de Saída | % | +/-3 |
| Eficiência do Módulo | % | 10,9 |
| Tensão na Potência Máxima | V | 17,3 |
| Corrente na Potência Máxima | A | 1,16 |
| Tensão do Circuito Aberto | V | 21,3 |
| Corrente de Curto Circuito | A | 1,32 |

**Fonte: Yingli Solar (2014).**

Na Tabela 4 são apresentadas as informações de dimensões, peso, e módulos da placa utilizada, também detalhadas no manual do fabricante.

**Tabela 4: Características Gerais da Placa YGE 20.**

| Características Gerais | | |
|---|---|---|
| **Descrição** | **Unidade** | **Valores** |
| Dimensões (Comprimento x Largura x Espessura) | mm | 525 x 350 x 25 |
| Peso | Kg | 2,58 |
| Número de Módulos por caixa | unidade | 6 |
| Dismensões da caixa (Comprimento x Largura x Espessura) | mm | 570 x 380 x 210 |

**Fonte: Yingli Solar (2014).**

O aumento da temperatura influencia negativamente as características da placa fotovoltaica como descrito anteriormente. Na Tabela 5 podem ser observadas as características térmicas do painel da série YGE 20, destacando que nesse caso a cada grau Celsius de elevação acima da temperatura nominal máxima, 46(+/- 2), há uma perda de 0,45% da potência.

**Tabela 5: Características Térmicas da Placa YGE 20.**

| Características Térmicas | | |
|---|---|---|
| **Descrição** | **Un** | **Valores** |
| Temperatura nominal de operação da célula | °C | 46 +/- 2 |
| Coeficiente de Temperatura da Pmax | %/°C | -0,45 |
| Coeficiente de Temperatura da Vca | %/°C | -0,37 |
| Coeficiente de Temperatura da Icc | %/°C | 0,06 |

**Fonte: Yingli Solar (2014).**

## 2.3. TIPOS DE RASTREADORES

Segundo Prinsloo e Dobson (2014), o seguidor ou rastreador solar é o dispositivo que orienta geradores solares de forma que a incidência de raios solares perpendiculares à superfície seja máxima. A trajetória solar pode ser obtida por cálculos onde são considerados parâmetros como declividade terrestre, latitude, ângulo horário, dia do ano, etc.

Levando em consideração todos os métodos de rastreadores solares, os seguidores podem ser agrupados em dois grupos principais, caracterizados pelo seu grau de liberdade, ou seja, quantos eixos de rotação possui, conforme Figura 7 (CHONG et al., 2014).

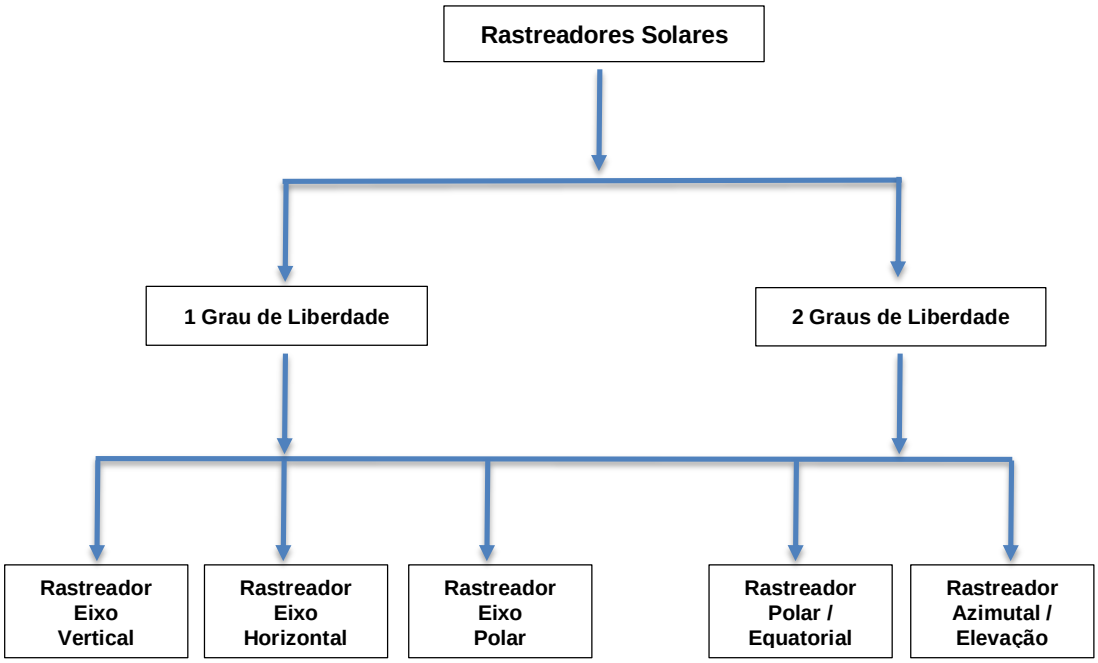



**Figura 7: Tipos de plataformas de rastreadores solares.**

**Fonte: Chong et al. (2014).**

Os principais tipos de rastreadores solares são descritos a seguir:

- Rastreador de eixo vertical: o eixo é co-linear com o eixo do zênite[6] (representado pelo eixo z na Figura 8) e é conhecido como rastreador de azimute[7] do sol;
- Rastreador de eixo horizontal: o eixo de rastreamento permanece orientado paralelo à superfície da terra, sendo assim, sempre orientado na direção leste-oeste ou norte-sul;
- Rastreador polar: o eixo de rastreamento é inclinado horizontalmente por um ângulo orientado ao longo do sentido norte-sul. (Exemplo: Rastreador solar de eixo inclinado de latitude).
- Rastreador azimutal elevação: possui dois eixos de movimentação, vertical e horizontal, proporcionando um grande rendimento da radiação solar entre os modelos disponíveis. Para realizar seu controle, as correções angulares são feitas pelo ângulo de azimute (horizontal, $\gamma$) e ângulo de elevação (vertical, $\alpha$), conforme demonstrado na Figura 8.

[6] Ponto de referência para observação do celestial.
[7] Medida de abertura angular que vai do norte geografico até a projeção de um alvo com o horizonte

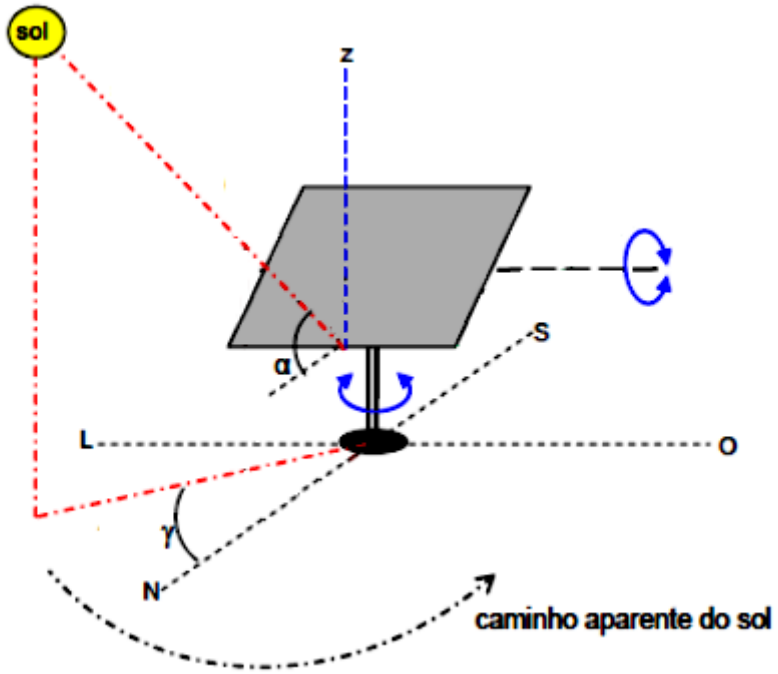



**Figura 8: Rastreador solar azimutal elevação.**
**Fonte: Prisloo e Dobson, 2014.**

Para o rastreador azimutal/elevação, a equação do ângulo de incidência direta é a mais simplificada dentro dos modelos conhecidos (Prinsloo & Dobson, 2014).

Os movimentos de rotação e translação da Terra ocasionam diferentes ângulos dos raios solares na superfície terrestre. A posição do sol para um referencial no planeta Terra depende da localização do ponto de interesse, do fuso horário e do dia no ano local.

Através do *Local Standard Time Meridian* (LSTM), usado como referência de fuso horário, é feita a compensação do fuso local pela Equação 1:

$$LSTM = 15°\Delta T_{GMT}, \quad (1)$$

onde $\Delta T_{GMT}$ é a diferença entre o fuso horário local com relação ao fuso horário do meridiano de Greenwich. Em seguida no cálculo da trajetória do sol é descriminada a equação do tempo (EoT), equação que empiricamente tem a função de corrigir a excentricidade da orbita terrestre e o eixo de elevação terrestre dada pela Equação 2,

$$EoT = 9{,}87\,\mathrm{sen}(2B) - 7{,}53\cos(B) - 1{,}5\,\mathrm{sen}(B), \quad (2)$$

onde $B = \left({}^{360}/_{365}\right)(d - 81)$ , em graus e $d$ é o número de dias do ano.

Para compensar as variações longitudinais dentro da mesma área de um fuso horário é incorporado o fator de correção (TC) conforme a Equação 3, levando em consideração a Equação 2.

$$TC = 4(Longitude - LSTM) + EoT, \quad (3)$$

onde o fator 4 que multiplica a equação é em virtude da rotação de 1 grau da terra a cada 4 minutos.

O horário solar *Local Solar Time* (LST) (Equação 4) é encontrado através do fator de correção calculado através da Equação 3 levando em consideração o *Local Time* (LT).

$$LST = LT + {}^{TC}/_{60} \quad (4)$$

Os valores calculados até o momento não são valores angulares. Para efetuar a conversão do valor *Local Solar Time* (LST) em valores angulares utiliza-se a Equação 5, onde HRA é o horário angular.

$$HRA = 15°(LST - 12) \quad (5)$$

Por definição, o ângulo horário é 0° no meio dia. A cada hora cheia a Terra se movimenta 15°, fazendo assim cada hora corresponder a uma porção de 15° no céu. Pela manhã corresponde a valores negativos e a tarde a valores positivos.

Em virtude da inclinação da Terra em seu próprio eixo de rotação surge o ângulo de declinação. O ângulo de declinação tem variação sazonalmente, ou seja, praticamente a cada estação do ano ele assume valores distintos. Apenas nos equinócios o ângulo de declinação é 0°. O ângulo de declinação $\delta$ é calculado segundo a Equação 6:

$$\delta = sen^{-1}(\mathrm{sen}(23{,}45°)\,\mathrm{sen}(\left({}^{360}/_{365}\right)(d - 81))), \quad (6)$$

onde $d$ é o dia do ano e o valor de 23,45° é devido aos solstícios de junho e dezembro, onde o ângulo de declinação recebe seus maiores valores de amplitude.

O ângulo de elevação, é o ângulo formado pela projeção da reta que liga um ponto especifico da Terra com um plano horizontal no mesmo ponto, o qual tem de 0° no nascimento do sol e 90° quando o sol está exatamente no ponto mais alto acima do dispositivo (configuração nos equinócios). O ângulo de elevação tem variação durante o dia todo, dependendo particularmente da latitude local e do dia do ano.

Para os diferentes hemisférios existe uma formula para adaptar o lado que o sol transcreve sua trajetória do referencial terrestre.

No hemisfério norte, a Equação 7 reproduz a correção da trajetória do ângulo de elevação do sol, $\alpha$. O mesmo ocorre no hemisfério sul, através da Equação 8.

$$\alpha = 90 - \varphi + \delta \tag{7}$$

$$\alpha = 90 + \varphi - \delta, \tag{8}$$

onde $\varphi$ é a latitude do local de interesse e $\delta$ é o ângulo de declinação.

O ângulo de elevação é calculado através da Equação 9:

$$\alpha = sen^{-1}[sen\delta\ sen\varphi\ +\ cos\delta\ cos\varphi \cos(HRA)], \tag{9}$$

onde HRA é o horário angular.

Para saber o horário que o dispositivo deve movimentar a placa, somente na parte produtiva do dia é calculado o horário que o sol nasce e o horário do pôr do sol. O nascimento do sol e o pôr do sol são calculados pelas Equações 10 e 11, respectivamente:

$$Sunrise = 12 - {}^{1}/_{15^{\circ}}\, cos^{-1}\left({}^{-sen\varphi\ sen\delta}/_{cos\varphi\ cos\delta}\right) - {}^{TC}/_{60} \tag{10}$$

$$Sunset = 12 + {}^{1}/_{15^{\circ}}\, cos^{-1}\left({}^{-sen\varphi\ sen\delta}/_{cos\varphi\ cos\delta}\right) - {}^{TC}/_{60}, \tag{11}$$

onde TC é correção do fuso horário.

O ângulo de azimute solar tem variação ao longo do todo o dia. Por definição, foi considerado no projeto que o ponto zero do dispositivo seria voltado para o leste, como a Figura 9 demonstra. O cálculo do ângulo azimute solar é dado pela Equação 12,

$$\gamma = cos^{-1}\left(\text{sen}\delta\cos\theta - \cos\delta\,\text{sen}\theta\cos(HRA)/_{cos\propto}\right), \quad (12)$$

onde $\propto$ é o ângulo de elevação, calculado na equação 9.

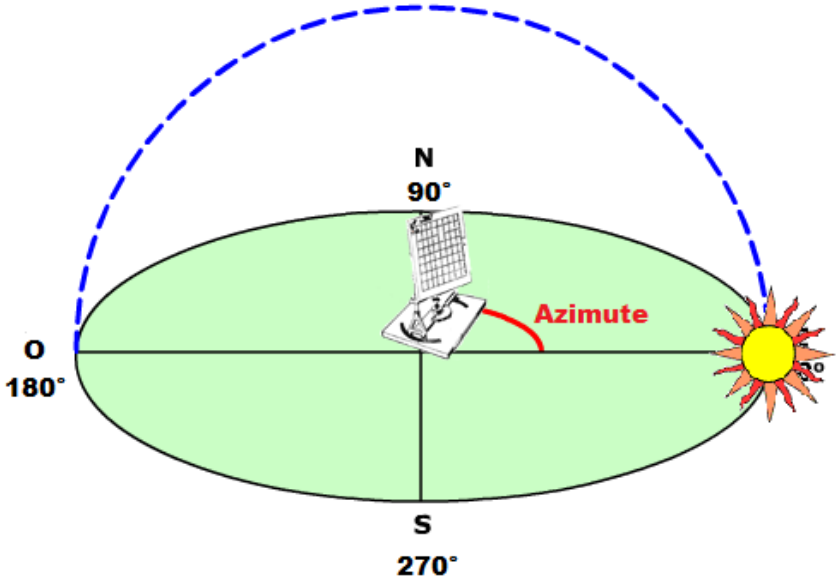



**Figura 9: Posição inicial do dispositivo.**

**Fonte: Autoria Própria.**

## 2.4 MOTORES

A escolha de um motor elétrico para determinada aplicação depende essencialmente das características da carga, dos motores elétricos e da forma como se deseja fazer o controle dessa carga (Stephan, 2013).

Em sistemas onde se deseja efetuar o controle de posição, a prioridade está na precisão do movimento, levando assim a escolha de um acionamento que possua confiabilidade em toda a trajetória da carga.

### 2.4.1 Motor *Brushless* DC com sensor *hall*

O princípio de funcionamento do motor *brushless* DC se dá através do alinhamento dos campos magnéticos. Na Figura 10 é ilustrado o estator, formado pelo enrolamento de fases, enquanto o rotor é formado por imãs permanentes.

Os imãs permanentes formam um campo magnético principal, enquanto as correntes aplicadas nas bobinas do motor criam um fluxo magnético que reagem com as dos imãs permanentes, a fim de alinhá-los, formando um conjugado eletromecânico (CODY, 2009).

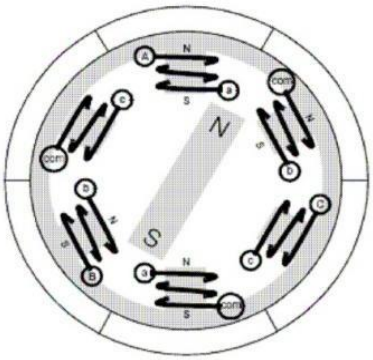



**Figura 10: Circuito simplificado Motor *Brushless* DC.**
**Fonte: Cody (2009).**

O sensor *hall* tem como função detectar a transição dos polos do motor gerando um sinal com formato de trem de pulso, conforme ilustrado na Figura 11. Com essa onda pode-se adquirir a velocidade e a posição do motor, efetuando a contagem dos pulsos. O passo mínimo considerado nessa situação é a contagem de um pulso gerado pelo sensor.

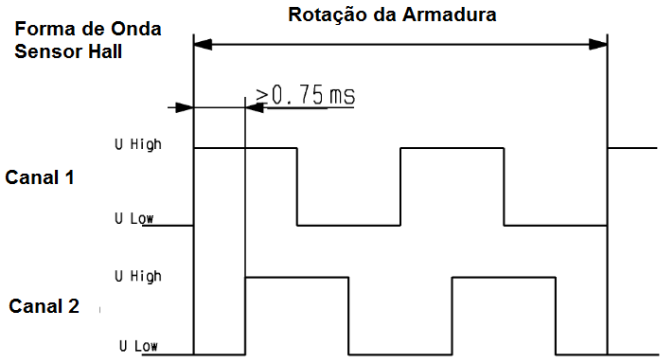



**Figura 11: Sinal gerado pelos sensores *hall* do motor.**
**Fonte: Manual do fabricante do motor (Anexo A)**

### 2.4.2 Servomotores

O controle do servomotor baseado em potenciômetro, ou *Radio Control Servo*, tem como princípio de funcionamento, demonstrado na Figura 12, o envio de um sinal de largura de pulso (PWM) pelo microcontrolador. Esse sinal é então traduzido para um valor de tensão (por exemplo, PWM com largura de 1 ms é convertido para uma tensão de 2,5 V), ou seja, uma instrução de posição. Ao receber o comando o servo irá verificar a posição atual, através da medição da tensão do potenciômetro acoplado ao seu eixo (se para o mesmo exemplo o potenciômetro está medindo 2,5 V). Caso exista diferença de tensão entre o comando do PWM e a atual posição, o motor irá acionar até zerar a diferença, ou seja, atingir a posição do comando (BLUE POINT ENGINEERING, 2007).

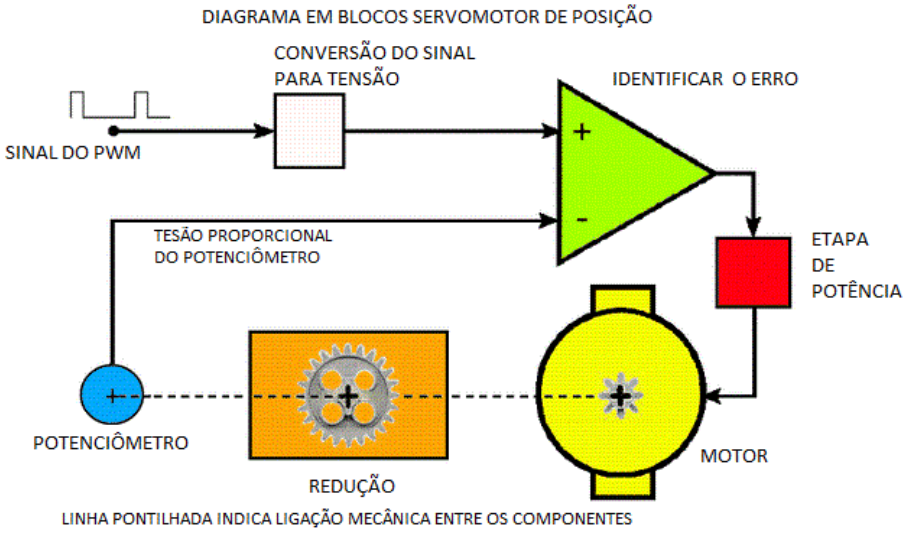



**Figura 12: Diagrama de funcionamento do RC servomotor.**
**Fonte: Autoria Própria.**

## 2.5 SENSORES DE LUMINOSIDADE

Sensores de radiação eletromagnética que tem faixa espectral do ultravioleta ao infravermelho, são chamados de sensores de luminosidade. A absorção de luz por parte de um material sensível a luminosidade pode causar dois tipos de modificação no material, modificação quântica ou térmica. Os sensores quânticos operam de uma faixa ultravioleta à infravermelho médio, enquanto um sensor térmico é usado para infravermelho médio a além da faixa espectral infravermelho (FRADEN, 2010).

### 2.5.1 Rastreadores Solares: Fotodiodos e Fototransistores

Aparelhos fotosensitivos e seus princípios de operação são amplamente usados em controles de malha fechada para sistemas de rastreamento solar. Nesses casos sensores luminosos e detectores infravermelhos podem ser empregados para automatizar o rastreamento solar ou para posicionamento de um painel.

Há projetos que utilizam duplo ângulo de rastreamento através do acoplamento de sensores para detectar se um painel solar está orientado para apontar diretamente o sol. Os fototransistores podem acionar diretamente o servo motor, comandando o motor que direciona o módulo fotovoltaico (PRINSLOO & DOBSON, 2014).

A Figura 13 mostra um típico diagrama do circuito com interface de fotodiodos em quadrantes que, associados a amplificadores operacionas, oferecem o sinal de erro direto para o acionamento dos motores (FRADEN, 2010).

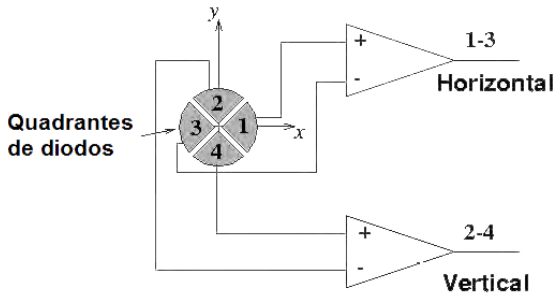



**Figura 13: Circuito para um fotodiodo quadrante.**
**Fonte: Prinsloo e Dobson (2014).**

### 2.5.2 Rastreamento Solar: Resistores Sensíveis a Luz

O *light-dependant-resistor* (LDR) opera no princípio de um fotocondutor no qual a resistência do semicondutor diminui à medida que a exposição à luz aumenta. O semicondutor absorve a energia luminosa, fazendo com que elétrons livres se movam sobre as lacunas do silício, diminuindo a resistência do LDR (PRINSLOO & DOBSON, 2014).

O fotoresistor é geralmente composto por uma capa plástica na qual existe uma lâmina de Sulfeto de Cádmio, que é uma substância sensível à luz. Assim, é um

componente passivo que converte energia luminosa em sinais elétricos ou fornece uma resistência específica.

Uma das vantagens do sensor é trabalhar com as mesmas intensidades luminosas que as do olho humano, permitindo sua operação com a luminosidade ambiente. Conforme mostra a Figura 14, a faixa de operação do LDR abrange desde a intensidade noturna até o brilho solar. Outras vantagens como a fácil aplicação do circuito (divisor de tensão) e a facilidade econômica foram decisivas para a utilização do LDR como o sensor  utilizado no projeto.

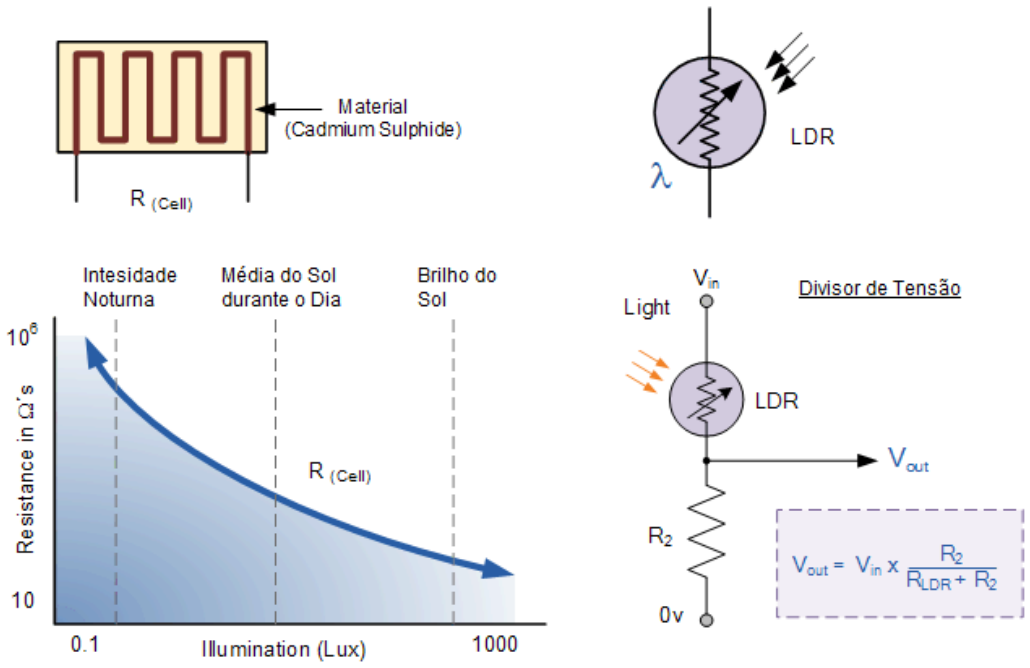



**Figura 14: Fotoresistência (LDR).**

**Fonte: Prinsloo e Dobson (2014).**

## 2.6 RASPBERRY PI

### 2.6.1 Características

A versão do Raspberry PI usada neste trabalho é a B+, pois é a versão que a equipe já possuia previamente.

Embora o Raspberry PI possua vários pinos de entrada e saída, nenhum deles tem a função de conversor analógico-digital e apenas um PWM.

Para acesso à rede, pode ser utilizado um adaptador Wi-Fi ou um cabo Ethernet para propiciar a troca de informações através de aplicação Web Server.

As configurações da rede serão realizadas manualmente, de forma que o servidor Web, acesso remoto SSH[8] e VNC[9] estejam sempre acessíveis através do endereço de IP 192.168.2.10.

[8] *Secure Shell*. Protocolo de rede criptografado que permite inicializar sessões em computadores remotos.
[9] *Virtual Network Computing*. Sistema de Compartilhamento de computação gráfica para controlar remotamente outro computador.

# 3. MATERIAS E MÉTODOS

Neste capítulo será apresentada a descrição completa do sistema de posicionamento da placa fotovoltaica para rastreamento solar via módulo controlador Raspberry PI para atender aos seguintes recursos:

- Captura de dados de potência gerada;
- Armazenamento em banco de dados;
- Servidor Web para extração dos dados;
- Comunicação via radiofrequência (Wi-Fi);
- Obtenção dos valores de sensores de luminosidade;
- Movimento de dois motores;
- Diferentes opções de posicionamento.

## 3.1 PLACA MICROPIC_DAELT (R0)

### 3.1.1 *Hardware* PIC

Apesar do Raspberry PI ser um módulo controlador de pequeno porte e com alto nível de processamento, a quantidade de pinos de entrada e saída (E/S) disponíveis, bem como as funcionalidades de saídas PWM e conversão analógica-digital são limitadas. A solução para a necessidade de seis conversores analógico-digitais (quatro para os sensores de luminosidade e duas para a medição de potência da placa fotovoltaica) e inicialmente de duas saídas PWMs para controlar os servo-motores, foi o uso da Placa MICROPIC_DAELT fornecida pelo orientador da equipe. A placa usa o microcontrolador PIC16F877A (Microchip Technology Inc, EUA), e foi desenvolvida pelo professor Dr. Amauri A. Assef e outros professores do DAELT ( Departamento de Eletrotécnica) para uso didático.

A placa MICROPIC_DAELT possui várias interfaces de saída, como *displays* LCD e 7 segmentos, além de proporcionar a aquisição de até 8 sinais analógicos e possuir 2 PWMs. Neste projeto a placa MICROPIC_DAELT apresenta as seguintes funcionalidades:

- Receber a ordem de movimento;
- Movimentar os motores;
- Receber os valores analógicos;
- Enviar as informações ao Raspberry PI.

### 3.1.2 *Firmware* do microcontrolador PIC16F877A

O *firmware* para o PIC foi desenvolvido usando o *software* mikroC e simulado com o *software* Proteus. O processo de funcionamento do programa é dependente de um comando externo inicial via interface de comunicação serial (USART - *Universal Synchronous Asynchronous Receiver Transmitter*), e está descrito abaixo:

1. Espera por mensagem;
2. Verifica se a mensagem está no formato descrito na Tabela 6, e pula para passo 8 caso contrário;
3. Move motor 1 dependendo do valor de pulsos recebido;
4. Envia via serial a confirmação de que o motor 1 foi movido;
5. Move motor 2 dependendo do valor de pulsos recebido;
6. Envia via serial a confirmação de que o motor 2 foi movido;
7. Obtém os valores dos pinos analógicos, e envia os valores via serial;
8. Envia via serial a confirmação de que o processamento foi finalizado;
9. Retorna ao passo 1.

**Tabela 6: Formato das Mensagens para o PIC.**

| Posição | Exemplo de Valor | Descrição |
|---|---|---|
| 1 | +,- | Sinal referente à direção do movimento do motor 1, pode ser positivo (+) ou negativo (-). |
| 2-4 | 142 | Valor em pulsos do movimento para o motor 1. |
| 5 | , | Separador. |
| 6 | +,- | Sinal referente à direção do movimento do motor 2, pode ser positivo (+) ou negativo (-). |
| 7-9 | 085 | Valor em pulsos do movimento para o motor 2. |
| 10 | $ | Símbolo que indica o final da mensagem, deve ser '$'. |

**Fonte: Autoria Própria.**

As mensagens geradas e enviadas via comunicação serial pelo microcontrolador estão descritas na Tabela 7:

**Tabela 7: Mensagens enviadas pelo PIC via módulo USART.**

| Estrutura | Exemplo | Utilização |
|---|---|---|
| ADC1::00XXXX | ADC1::000145 | Sensor de Luminosidade 1. O valor pode variar de 0 a 1023. |
| ADC2::00XXXX | ADC2::000025 | Sensor de Luminosidade 2. O valor pode variar de 0 a 1023. |
| ADC3::00XXXX | ADC3::000368 | Sensor de Luminosidade 3. O valor pode variar de 0 a 1023. |
| ADC4::00XXXX | ADC4::001010 | Sensor de Luminosidade 4. O valor pode variar de 0 a 1023. |
| ADC5::00XXXX | ADC5::000054 | Captura da tensão na placa fotovoltaica. O valor pode variar de 0 a 1023. |
| ADC6::00XXXX | ADC6::000751 | Tensão que representa a corrente na placa fotovoltaica. O valor pode variar de 0 a 1023. |
| POS1:[SINAL]000XXX | POS1:-000025 | Valor movimentado em pulsos do motor 1. |
| POS2:[SINAL]000XXX | POS2:+000102 | Valor movimentado em pulsos do motor 2. |
| END000000000 | END000000000 | Confirmação de final de processamento. |

**Fonte: Autoria Própria.**

O código utilizado para gerar o arquivo .hex que é então gravado no microcontrolador.

## 3.2 ESTRUTURA DE SENSORES

O fotoresistor (LDR) é um tipo de sensor resistivo cuja resistência diminui de acordo com o aumeno do nível de luz que incide sobre ele. Portanto, a medição da tensão enviada para o microcontrolador é feita sobre o resistor em série no circuito, de forma a obter relação direta com a luminosidade. A Figura 15 demonstra o circuito elétrico do sistema de sensores.

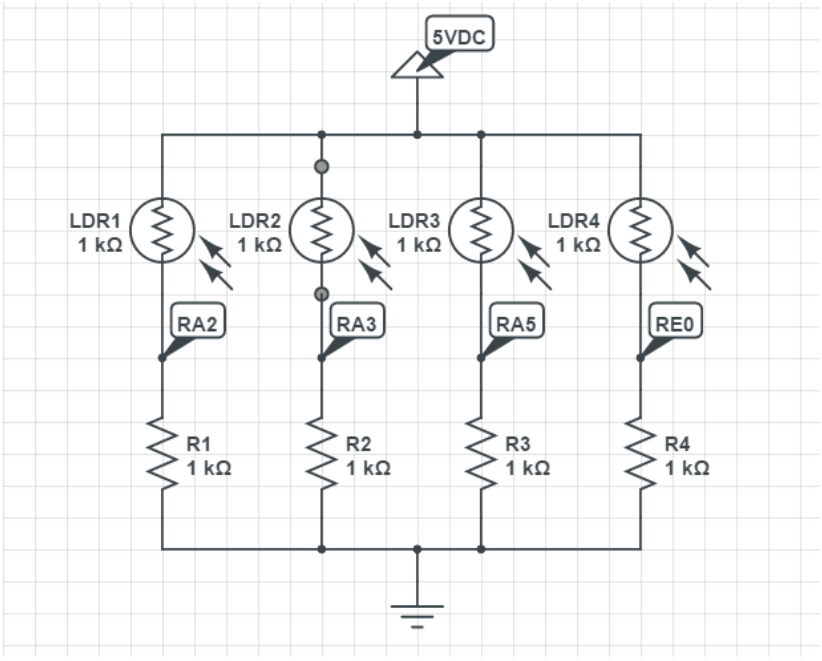



**Figura 15: Circuito Elétrico dos sensores de luminosidade.**
**Fonte: Autoria Própria.**

Onde RA2, RA3, RA, RE0 são as entradas analógicas do microncrolador PIC16F877A.

Na aplicação de rastreamento solar, os sensores são posicionados na base e separados por uma barreira em formato de cruz (Figura 16). A variação na resistência se dá através da sombra causada pela estrutura. Na posição ideal os quatro fotoresistores devem apresentar valores de tensão próximos. Os sinais de resposta são usados para determinar os erros nos ângulos, direcionando o rastreamento solar (PRINSLOO & DOBSON, 2014).

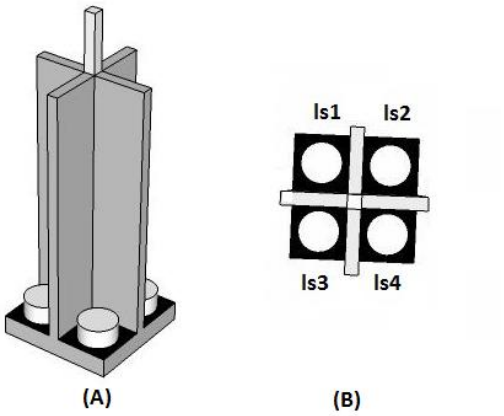



**Figura 16: (A) Vista frontal do conjunto de sensores de luminosidade (B) Vista superior do conjunto de sensores de luminosidade.**
**Fonte: Autoria Própria.**

Os valores de resistência ligadas em série com cada LDR foram encontrados através da análise da curva gráfica (Figura 17) do resistor pela luminosidade, a fim de obter maior utilização da faixa de entrada analógica do microcontrolador.

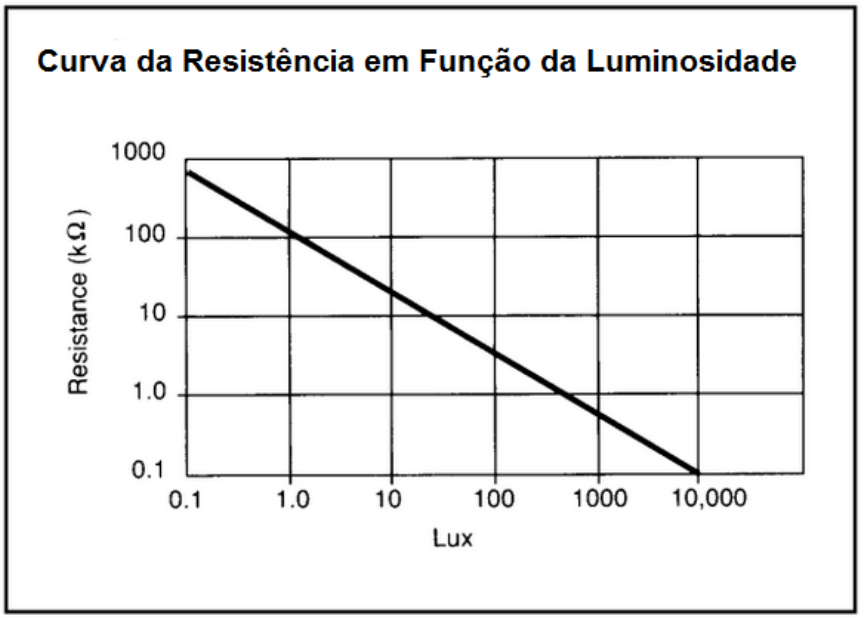



**Figura 17: Curva da resistência por luminosidade (LDR).**
**Fonte: Manual do Fabricante SILONEX.**

Pela Figura 17, observa-se que a resistência mínima do sensor é de cerca de 0,1 kΩ, e a resistência máxima é de aproximadamente 1000 kΩ.

A resistência escolhida de 1 kΩ foi determinada pelos alcances máximo e mínimo com o objetivo de abranger toda a faixa de trabalho das entradas analógicas.

Os alcances máximo e mínimo previstos para os sensores, alimentados em 5 V, são apresentadas nas Equações 13 e 14, respectivamente:

$$A_{máx} = 5\,[{}^{1x10^3}/_{(1x10^3 + 0{,}1x10^3)}] = 4{,}5\ V \quad (13)$$

$$A_{min} = 5\,[{}^{1x10^3}/_{(1x10^3 + 1000x10^3)}] = 0{,}9\ mV \quad (14)$$

O sistema de sensores implementado no projeto pode ser observado na Figura 18.

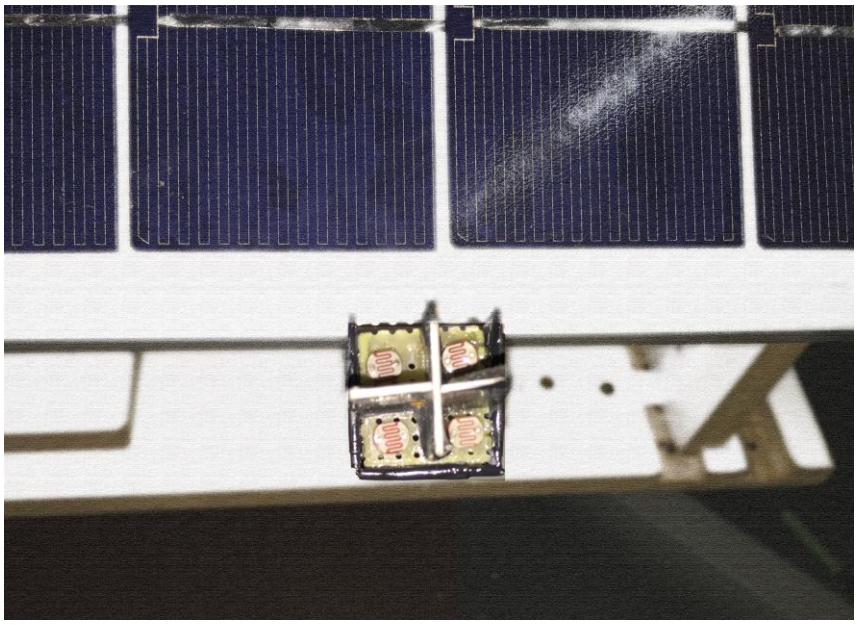


**Figura 18: Sistema de sensores utilizado.**
**Fonte: Autoria Própria.**

## 3.3 MOTORES

Inicialmente o projeto considerou utilizar servomotores *Radio Control Servo*, pelo seu *feedback* de posição e baixo custo se tratando de servomotores. O seu princípio de funcionamento utiliza o sinal de PWM de 20 ms que era gerado pelo microcontolador, porém a faixa para posicionamento do motor era somente entre 1 ms e 2 ms. Para a aplicação desejada a precisão apresentada pelo motor não foi satisfatório, dificultando o controle da placa.

A solução foi utilizar dois motores de corrente contínua devido a disponibilidade pela equipe e por possuírem sensores *hall* acoplados, possibilitando o controle da posição. O modelo FPG2 é produzido e comecializado pelo grupo Bosch e é utilizado em vidros elétricos automotivos. A representação do motor é ilustrada na Figura 19.

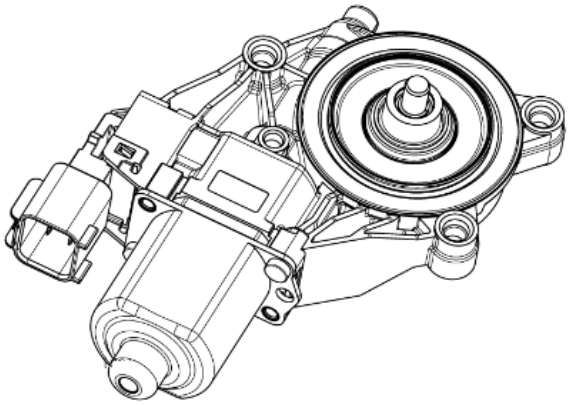


**Figura 19: Representação do motor utilizado no posicionamento das placas.**
**Fonte: Manual do fabricante do motor - Anexo A (2015).**

O sinal do sensor *hall* é alimentado com 5 V entre os pinos 3 e 6 do conector de saída do motor, conforme apresentado na Figura 20. Também são identificados os demais sinais de controle do motor (Figura 20).

| PINOS DE SAÍDA | |
|---|---|
| PIN | FUNÇÃO |
| 1 | CONTATO DO MOTOR 1 |
| 2 | CANAL 2 |
| 3 | ALIMENTAÇÃO SENSOR HALL |
| 4 | CONTATO DO MOTOR 2 |
| 5 | CANAL 1 |
| 6 | GND SENSOR HALL |

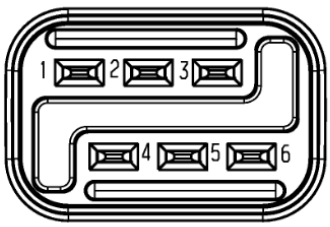



**Figura 20: Pinos de saída do motor.**
**Fonte: Manual do fabricante do motor - Anexo A (2015).**

O circuito auxiliar é composto por um resistor no pino do conector ligado ao motor em que se deseja obter a forma de onda, como mostrado na Figura 21. O uso dos pinos 2 e 5 se justificam caso ocorra necessidade de verificação do sentido de giro do motor, devido a defasagem dos sinais dos sensores.

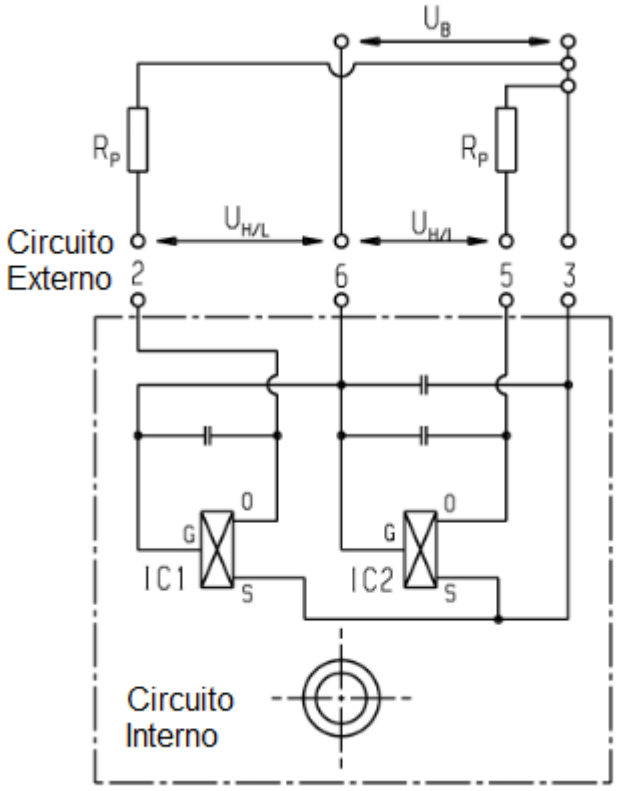



**Figura 21: Representação elétrica dos sensores hall.**
**Fonte: Manual do fabricante do motor - Anexo A (2015).**

Sendo:

- IC1 e IC2 são os sensores *hall* do circuito interno;
- $U_B$ é uma tensão de 5 V no terminal de alimentação dos sensores;
- $R_p$ é a resistência adotada, conforme o manual do fabricante do motor é estabelecido em $100\ \Omega$;
- $U_{H/L}$ é o sinal da forma de onda;
- $U_H$ nível lógico alto;
- $U_L$ nível lógivo baixo.

Para garantir o posicionamento, o sinal $U_{H/L}$ de cada motor é ligado a um réle da placa MICROPIC_DAELT. O réle comuta dependendo de qual motor está ligado no momento para fazer o controle de pulsos através do pino de interrupção do microcontrolador PIC16F877A. Dessa forma é possível contar os pulsos efetuados por motor individualmente. Quando o número de pulsos chega ao valor recebido via USART, o motor é desenergizado e o microcontrolador responde também via USART o fim do *loop*.

As especificações do manual do fabricante relatam que o motor suporta um impulso máximo de 39 V durante 400 ms e corrente nominal de 10 A. Sua operação normal é em 12 V, podendo trabalhar em tensões abaixo desse nível. Para o projeto é adotado alimentação com 3 V, pois o desempenho em questão de velocidade do movimento da placa foi adequada com o objetivo do projeto.

Para acionamento de cada motor foi utilizada uma placa *shiled* BTE13-003 com dois relés. Além disso, foi utilizada a placa MICROPIC_DAELT versão 0 como interface entre o módulo Raspberry PI e as duas placas *shields*.

O módulo BTE13-003 possui dois relés que são comandados através dos pinos IN1 e IN2, de forma que seja possível selecionar o sentido de rotação do motor. Essa configuração possibilita o acionamento individual dos motores com controle de sentido, conforme apresentado na Tabela 8.

**Tabela 8: Tabela verdade módulo BTE13-003.**

| IN1 | IN2 | Rotação |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | SH |
| 0 | 1 | SAH |
| 1 | 1 | 0 |

**Fonte: Autoria Própria.**

Pode-se observar que as saídas têm resposta *or exclusive* (XOR), que produz 0 como saída (não aciona o motor) se IN1 e IN2 são iguais caso contrário energiza os motores, sendo SH sentido horário e SAH sentido anti-horário.

Na Figura 22 é apresentado o esquema de conexão entre a placa MICROPIC_DAELT e os dois módulos BTE13-003.

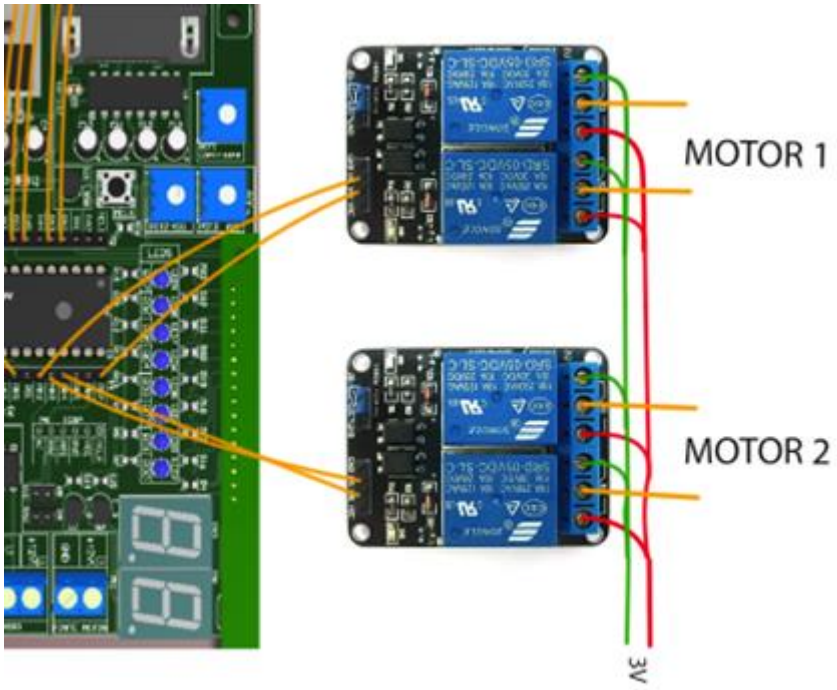



**Figura 22: Conexão dos módulos duplos relé BTE13-003.**

**Fonte: Autoria Própria.**

## 3.4 COMUNICAÇÃO ENTRE RASPBERRY PI E PIC16F877A

Para a troca de informações entre o PIC16F877A (Figura 23) e o Raspberry PI, foi escolhido o modo de comunicação USART (serial) por estar presente em ambas interfaces.

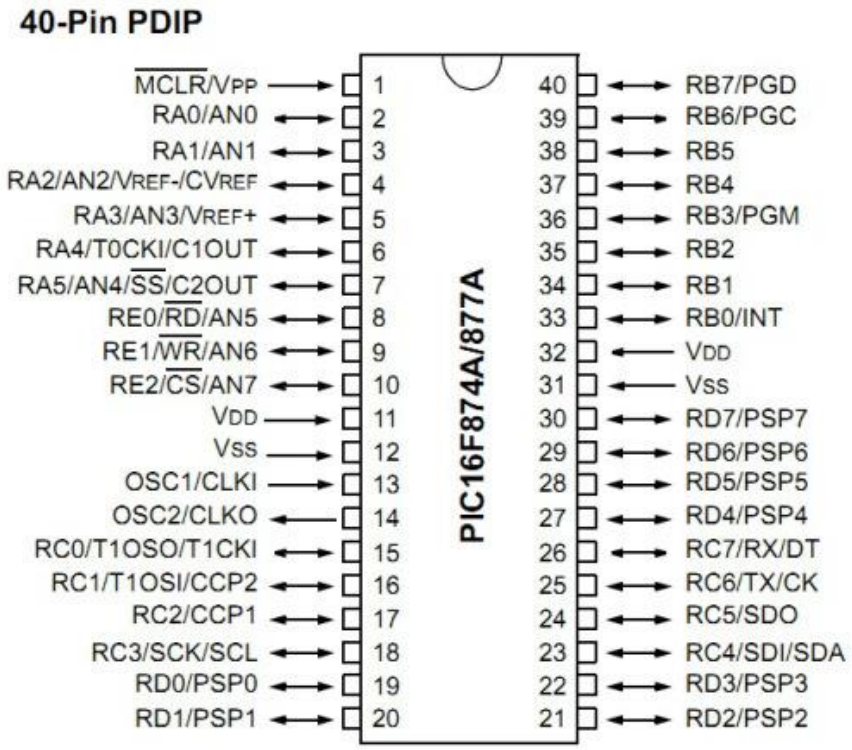



**Figura 23: Pinos de Saída e Entradas do PIC16F877A.**

**Fonte: Manual do fabricante Microchip (2015).**

Foram utilizados os pinos RC6 para transmissão e RC7 para recebimento de dados no PIC, que trabalha com um nível de tensão de 5V. Já no Raspberry PI, os pinos de transmissão e recebimento são os 08 e 10, respectivamente, mas com tensão de 3,3V. Para o envio a partir do Raspberry, a tensão de 3,3V foi suficiente para o recebimento do PIC, pois já são perceptíveis os níveis lógicos alto e baixo. No envio do sinal do PIC para o Raspberry PI foi necessário um divisor resistivo para ajustar a tensão a um nível próximo de 3,3V. O divisor resistivo é demonstrado na Equação 15,

$$3{,}3 = 5.\frac{R_a}{(R_a+R_b)}, \tag{15}$$

sendo $R_a$ a resistência em paralelo na saída para o Raspberry PI.

Portanto a relação entre as resistências é dada pela Equação 16,

$$R_a \cong 1{,}9411.R_b \tag{16}$$

O divisor resistivo escolhido, utilizando valores comerciais de resistências, está ilustrado na Figura 24, onde $R_a$ é a resistência de 47 kΩ e $R_b$ é a resistência em série 4,7 kΩ + 22 kΩ:

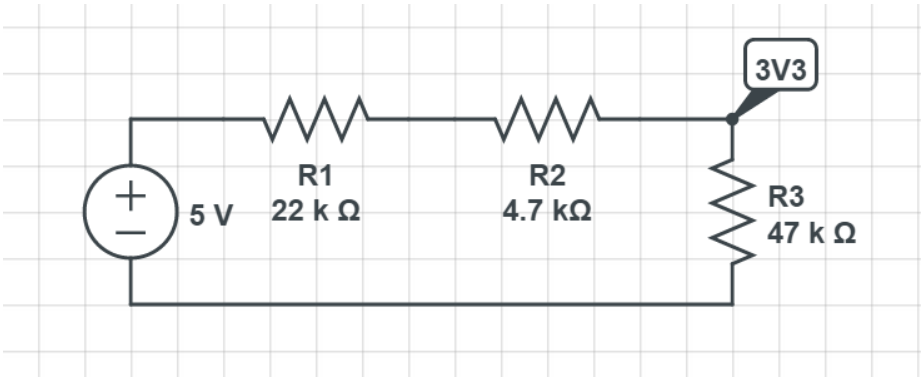



**Figura 24: Divisor de tensão USART.**

**Fonte: Autoria Própria.**

O valor calculado da tensão de entrada para o pino de recebimento USART do Raspberry PI sobre o resistor R3 é mostrado na Equação 17.

$$V_{3v3} = \left(\frac{5}{22.10^3.4{,}7.10^3.47.10^3}\right).47.10^3 \cong 3{,}2\ V \quad (17)$$

## 3.5 OBTENÇÃO DE POTÊNCIA DA PLACA FOTOVOLTAICA

De acordo com as informações da placa fotovoltaica, a tensão máxima que pode ser atingida é de 21,3 V.

Caso um resistor de 50 Ω seja ligado à placa em tensão máxima, a potência observada será de aproximadamente à calculada na Equação 18.

$$P = \frac{21{,}3^2}{50} = 9{,}074\ W \quad (18)$$

O resistor comercial mais próximo do caso acima é o de 47 Ω, que com um resistor *Shunt* de 1 Ω somam 48 Ω, ou seja, para a mesma tensão considerada acima tem-se a potência calculada através da Equação 19.

$$P = \frac{21{,}3^2}{48} = 9{,}452\ W \quad (19)$$

Os valores de corrente, tensão e potência no resistor *Shunt* estão demonstrados nas Equações 20, 21 e 22, respectivamente.

$$I_{shunt} = I = \frac{21,3}{48} = 0,444\ A \quad (20)$$

$$V_{shunt} = 0,444\ .\ 1 = 0,444\ V \quad (21)$$

$$P_{shunt} = 0,444^2 = 0,197\ W \quad (22)$$

Portanto, foi escolhida a resistência de 47 Ω na potência comercial de 10 W. O *Shunt* utilizado foi de 1 Ω na potência comercial de 10 W pelo fato da equipe já ter posse do componente, embora possam ser usados resistores de menor potência.

A medição da tensão no *Shunt*, como sua resistência é de 1 Ω, representa a corrente que passa pelo conjunto.

Para que a tensão da placa seja obtida, um divisor resistivo deve ser utilizado. A relação de divisão deve ser maior que 4,26, conforme a Equação 23:

$$\frac{(R1+R2)}{R1} = \frac{21,3}{5} = 4,26 \quad (23)$$

Para atingir a relação e ao mesmo tempo não impactar no resto do circuito, foram escolhidos os resistores R1=10 kΩ e R2=47 kΩ. Portanto, a tensão obtida no resistor de 10 kΩ deve ser multiplicada por 5,7 (Equações 24 e 25) para que o valor obtido fique próximo do real.

$$V_{out} = V_{in}.\frac{(R1+R2)}{R1} = \frac{57.10^3}{10^4} \quad (24)$$

$$\frac{V_{out}}{V_{in}} = 5,7 \quad (25)$$

No esquema apresentado na Figura 25, é possível observar o circuito com a carga utilizada, onde as aquisições dos valores de tensão serão usados para determinar a potência de saída da placa.

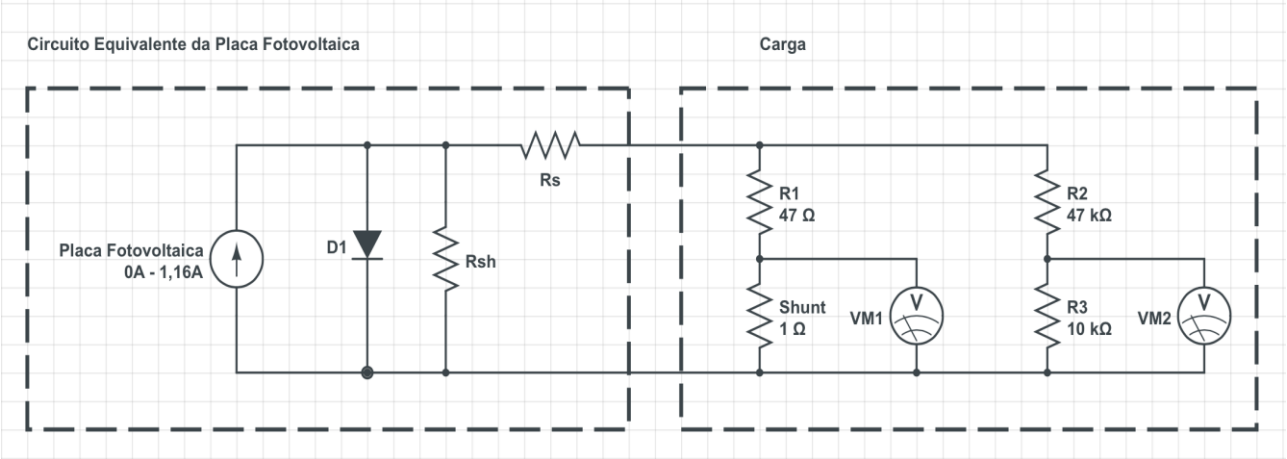



**Figura 25: Circuito Equivalente com a Carga Utilizada.**
**Fonte: Andrade (2006).**

A Figura 26 mostra o circuito de obtenção de corrente e tensão da placa fotovoltaica. Os pinos da esquerda são de alimentação provenientes da placa. Os pinos da direita são, em ordem, GND, medição de tensão no *Shunt* (V1) e medição de tensão da placa (V2).

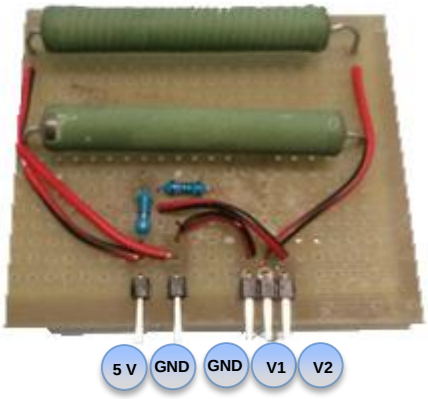



**Figura 26: Circuito Implementado da Carga da Placa Fotovoltaica.**
**Fonte: Autoria Própria.**

É importante observar que como os resistores não são de precisão (10%), devem existir certas divergências nos resultados práticos.

## 3.6 CONEXÃO DE ELEMENTOS DO SISTEMA

Para a obtenção dos valores analógicos, foram escolhidos os pinos RA2, RA3, RA5 e RE0 do microcontrolador PIC para os sensores de luminosidade mostrados no capítulo 2. Os pinos RE1 e RE2 são usados para a obtenção do valor de potência gerada pela placa fotovoltaica. Os pinos RA0 e RA1 não foram escolhidos por estarem conectados a *trimpots*.

O movimento dos motores DC está relacionado aos pinos RB2 e RB7 do microcontrolador para os pinos IN1 e IN2 do relé do motor 1, respectivamente. Os pinos RB4 e RB5 da mesma forma são ligados aos pinos IN1 e IN2 do relé do motor 2, respectivamente.

A obtenção da precisão do movimento efetuado pelo motor se dá através dos sensores *hall* ligados ao pino RB0, que é uma entrada de interrupção do microcontrolador, possibilitando que cada um dos pulsos gerados sejam contabilizados. Quando o número de pulsos chega ao valor solicitado pelo Raspberry PI, o motor para e o PIC responde positivamente à solicitação.

O pino RC2 é usado como saída digital para a ativação do relé que está localizado na própria placa MICROPIC_DAELT. O relé tem como função a seleção do sensor que deve ser ligado ao pino RB0 de interrupção, visto que o PIC16F877A possui apenas um pino de interrupção externa para borda de subida.

A Figura 27 demonstra as conexões que envolvem o microcontrolador PIC.

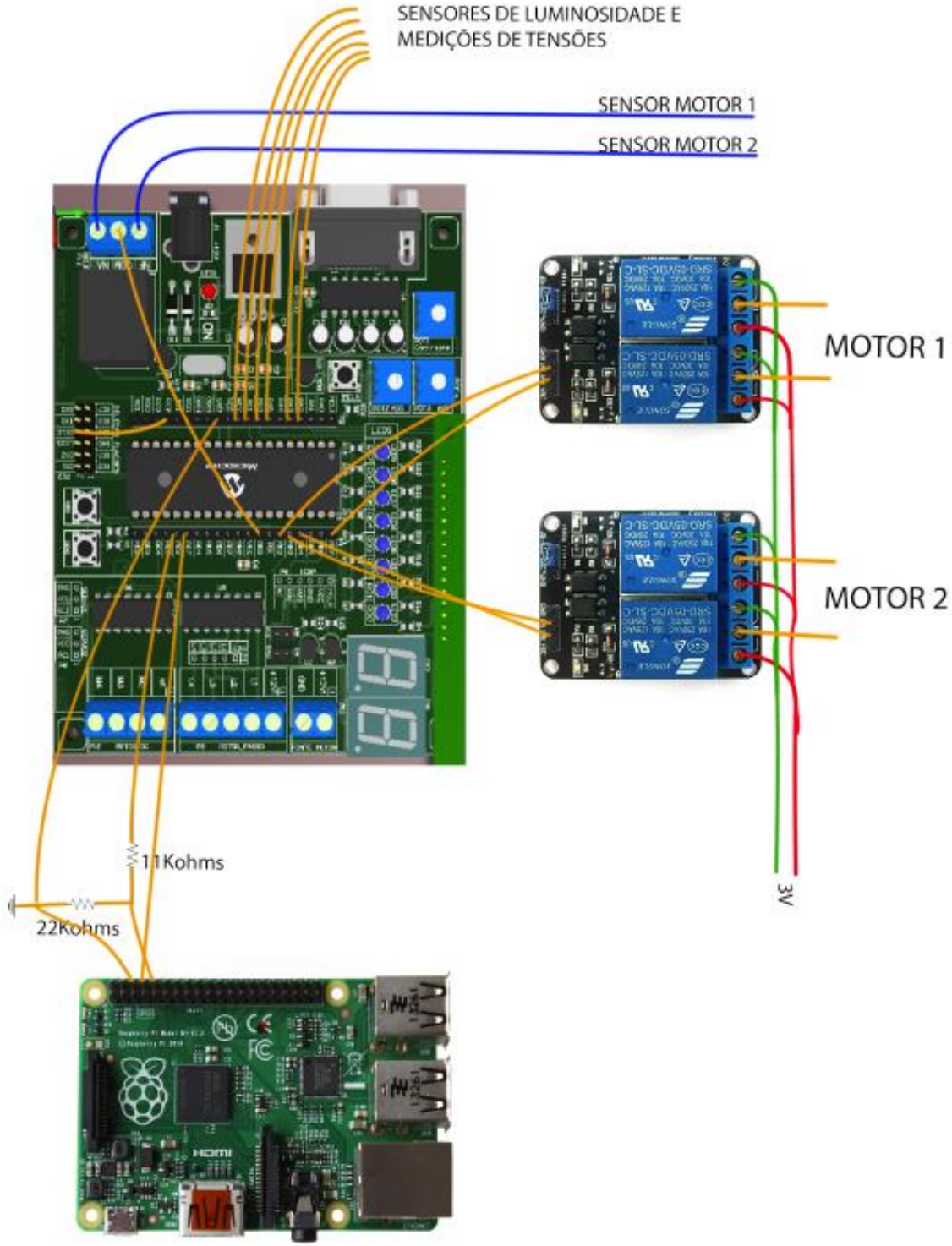



**Figura 27: Ligação elétrica do Projeto.**

**Fonte: Autoria Própria.**

Na Figura 28 é apresentada a placa auxiliar usada para facilitar as conexões de alimentação e interfaceamento. É importante destacar que esta possui uma alimentação externa à alimentação da placa do microcontrolador, de modo que não sobrecarregue o sistema, pois não há pinos suficientes para todas as interfaces.

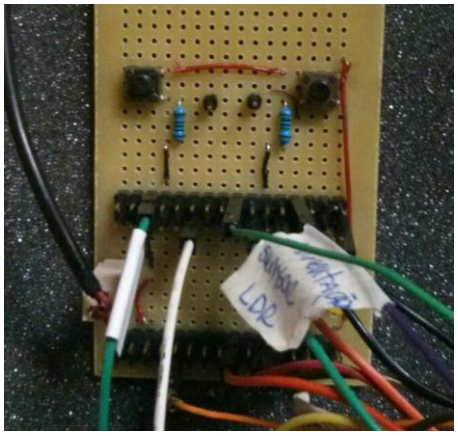


**Figura 28: Placa auxiliar para ligações.**
**Fonte: Autoria Própria.**

## 3.7 *RASPBERRY PI*

### 3.7.1 Configuração do Raspberry PI

Antes de começar a desenvolver o sistema de rastreamento solar, foram necessárias algumas configurações prévias para deixar o Raspberry PI pronto para funcionamento.

Primeiramente, como sistema operacional, foi escolhido o Raspbian, por ser o sistema recomendado no *site* oficial do Raspberry PI. Foram então alteradas as configurações de rede, para que os endereços de IP independentemente da conexão sejam fixos, proporcionando uma maior facilidade de acesso tanto para desenvolvimento do sistema como para interação com o servidor Web. As configuraçoes de rede são as seguintes:

Via cabo:

- IP Fixo 192.168.2.10;
- Máscara de rede: 255.255.255.0;
- Gateway: 192.168.2.1.

Via Wi-Fi:

- IP Fixo 192.168.2.11;
- Máscara de rede: 255.255.255.0;
- Gateway: 192.168.2.1.

Em seguida, os seguintes *softwares* foram instalados:

- Apache2 - Servidor web;

- PHP5 - Linguagem de programacao para *websites*;
- libapache2-mod-php5 - Biblioteca para funcionamento do PHP5 com o Apache;
- MySQL - Banco de dados;
- php-mysql - Biblioteca para integração do MySQL com o PHP5;
- Python 2.7 - Linguagem de programação a ser utilizada no sistema;
- MySQL-python - Biblioteca para integração entre MySQL e Python;

Com isto, o ambiente já está preparado para a criação do sistema.

### 3.7.2 Banco de dados

Após a instalação do MySQL, o banco de dados está pronto para ser configurado. Para acessar o servidor do banco, pode ser utilizado o usuário "*root"* com a senha "raspberry".

A Tabela 9 descreve cada uma das variáveis do sistema.

**Tabela 9: Descrição das variáveis do sistema.**

| Item | Tipo | Descrição |
|---|---|---|
| id | int | Índice para as linhas da tabela |
| power | float | Potência da placa fotovoltáica obtida |
| voltage | float | Tensão da placa fotovoltáica obtida |
| current | float | Corrente da placa fotovoltáica obtida |
| pos1 | float | Ângulo de posição referente ao motor 1 |
| pos2 | float | Ângulo de posição referente ao motor 2 |
| ls1 | float | Valor do sensor de luminosidade 1 [V] |
| ls2 | float | Valor do sensor de luminosidade 2 [V] |
| ls3 | float | Valor do sensor de luminosidade 3 [V] |
| ls4 | float | Valor do sensor de luminosidade 4 [V] |
| opmode | int | Modo de operação |
| date | date | data da captura |
| time | time | horário da captura |

**Fonte: Autoria Própria.**

No MySQL, foi definido o nome do banco a ser usado como "tccdb". E ele é composto por apenas uma tabela, "suntrackerdata".

### 3.7.3 Servidor Web e extração de dados

A partir da estrutura do banco de dados criado acima, uma página web foi desenvolvida para que os dados sejam obtidos e exportados. Os formatos de saída implementados são xml, json e csv. O acesso à página é feito a partir do próprio IP do Raspberry PI, utilizando a porta padrão 80.

Toda a seleção de parâmetros é realizada na própria barra de endereços do navegador, em que devem ser passados parâmetros para as buscas no banco de dados. Por exemplo, ao utilizar o endereço

http://192.168.2.10/index.php?mode=xml&select=pos1,pos2,time&opmode=2&days=4&btime=21:14:27 ,estão sendo usados os parâmetros:

- mode, com o valor "xml";
- select, com o valor "pos1,pos2,time";
- opmode, com o valor "2";
- days, com o valor "4";
- btime, com o valor "21:14:47".

O resultado obtido deve ser no formato xml e conter os valores de posição 1, posição 2 e o horário da captura nos quais o modo de operação seja o de número 2, dos últimos quatro dias, e em que a hora seja maior que 21:47:47. A Figura 29 mostra a saída gerada.

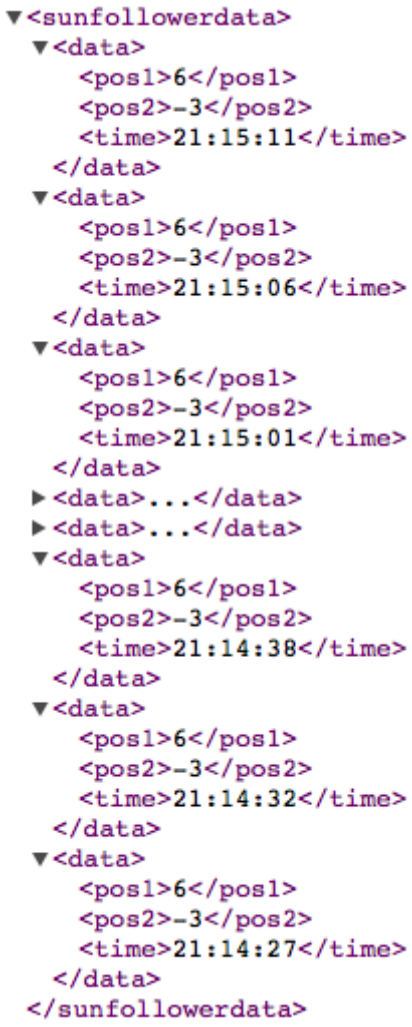



**Figura 29: Exemplo de extração dos dados no formato json.**

**Fonte: Autoria Própria.**

O formato do endereço eletrônico local para as pesquisas deve seguir o modelo a seguir:

http://[IP]/index.php?[PARAMETRO]=[VALOR]&[PARAMETRO]=[VALOR]&...&[PARAMETRO]=[VALOR]

Os parâmetros para as buscas não possuem ordem no endereço e estão listados na Tabela 10.

**Tabela 10: Parâmetros de busca via endereço eletrônico.**

| Parâmetro | Descrição |
|---|---|
| select | Seleciona os campos a serem retornados pela busca, e devem ser escritos entre vírgulas. Podem ser:<br>• power;<br>• voltage;<br>• current;<br>• pos1;<br>• pos2;<br>• ls1(ls2,ls3,ls4);<br>• opmode;<br>• date;<br>• time.<br>Ex.; "pos1,pos2,ls1,ls2,ls3,ls4,date,time". |
| minpos1 | Busca apenas os resultados que possuam o valor de posição do motor 1 maiores que o parametrizado. |
| maxpos1 | Busca apenas os resultados que possuam o valor de posição do motor 1 menores que o parametrizado. |
| minpos2 | Busca apenas os resultados que possuam o valor de posição do motor 2 maiores que o parametrizado. |
| maxpos2 | Busca apenas os resultados que possuam o valor de posição do motor 2 menores que o parametrizado. |
| opmode | Busca apenas resultados que possuam o modo de operação escolhido. |
| days | Busca apenas os resultados mais recentes que o número parametrizado de dias a contar do dia atual. |
| hours | Busca apenas os resultados mais recentes que o número parametrizado de horas a contar da hora atual. |
| mode | Seleciona o modo de saída. Pode ser:<br>• json;<br>• xml;<br>• excel. |
| bdate | Busca apenas resultados mais recentes que a data parametrizada. |
| edate | Busca apenas resultados mais antigos que a data parametrizada. |
| btime | Busca apenas resultados mais cujo horário seja maior que o horário parametrizado. |
| etime | Busca apenas resultados mais cujo horário seja menor que o horário parametrizado. |

**Fonte: Autoria Própria.**

Caso o formato de saída não seja especificado, o formato de saída padrão será o JSON, e caso não sejam selecionados os campos de saída, todos serão usados.

Os arquivos PHP utilizados para o Web Server são o dbconnect.php e index.php, e estão localizados na pasta /var/www/ do Raspberry PI.

O arquivo dbconnect.php faz a conexão com o banco de dados, enquanto o arquivo index.php recebe a solicitação e efetua as buscas no banco. Os códigos se encontram no APÊNDICE H.

### 3.7.4 *Software* Principal

O programa, como descrito anteriormente, deve movimentar os motores, estabelecer uma forma de configuração remota do modo de funcionamento e permitir a extração dos dados obtidos.A linguagem de programação utilizada foi Python, pois além do grupo já ter familiaridade, tem uma boa eficiencia de uso de memoria e processamento.Outra ideia utilizada foi a de fazer um sistema modular, ou seja, que seja construído um sistema descentralizado, em que cada setor ou módulo funcione independentemente um do outro. Uma grande vantagem deste tipo de sistema é que problemas podem ser encontrados mais facilmente, além de permitir que tarefas em paralelo sejam realizadas e controladas mais facilmente.

Foram definidas algumas características que cada módulo deve obedecer de forma a funcionar corretamente, listadas a seguir:

- Os módulos devem estar localizados na mesma pasta do módulo central (main.py), e possuir extensao .py;
- Devem possuir as funções start (para iniciar o módulo), stop (para parar o módulo) e check (para verificar a atual situação do módulo);
- Devem possuir uma classe *mainThread* com duas funções, *__init__* e run. Esta é a classe que irá ser iniciada de forma que o módulo seja executado em um processo paralelo (*thread*);
- Um modelo para novos módulos pode ser visto em APÊNDICE A.

#### 3.7.4.1 Módulo Main.py

É o módulo central do programa, que tem como função importar todos os outros módulos e colocá-los em execução. Além disso, é nele que se encontram definidas todas as variáveis chave do sistema:

- *power* - Valor de potência instantânea da placa fotovoltaica;
- *voltage* - Valor de tensão instantânea da placa fotovoltaica;
- *current* - Valor de corrente instantânea da placa fotovoltaica;
- ls1 - Valor instantâneo do sensor de luminosidade 1;
- ls2 - Valor instantâneo do sensor de luminosidade 2;
- ls3 - Valor instantâneo do sensor de luminosidade 3;
- ls4 - Valor instantâneo do sensor de luminosidade 4;
- pos1 - valor da posiçao atual do motor 1 da placa fotovoltaica (vertical);
- pos2 - valor da posiçao atual do motor 2 da placa fotovoltaica (horizontal);
- *desiredpos1* - próximo valor desejado para pos1;
- *desiredpos2* - próximo valor desejado para pos2;
- *opmode* - modo de operação do sistema, pode ser:
  - 0 - Somente haverá alteração da posição da placa por operação manual;
  - 1 - A posição será alterada de forma automática a partir dos valores dos sensores de luminosidade;
  - 2 - A posição será alterada de forma automática a partir de modelo de posição do sol, considerando o dia do ano e a hora do dia.
- *outputmodules* - é a variável que armazena quais módulos devem receber as informações sobre o sistema (*logs*).

Funções de outros módulos podem ser executadas utilizando a função *cmdhandler* que se encontra em Main.py. Esta função é chamada quando algo é digitado na própria tela em que foi iniciado o programa e quando um comando é recebido através de interface externa através do módulo TCPcom.py, como será mostrado no subtópico 3.7.4.4.

A estrutura na qual a função reconhece os comandos é a seguinte:

[Nome_do_módulo]_[Função_no_módulo]_[Parâmetro_1]_[Parâmetro_2]_..._[Parâmetro n].

Portanto, como anteriormente foi definido que todo módulo possui uma função *stop*, de parada do módulo, pode-se, por exemplo, parar qualquer módulo ao digitar: [Nome do módulo]_stop.

A lista com as funções que podem ser executadas se encontram na Tabela 11, e o código fonte no APÊNDICE C.

**Tabela 11: Funções que podem ser utilizadas no sistema.**

| Comando | Descrição |
|---|---|
| dbcom_start | Inicia o módulo dbcom. |
| dbcom_stop | Para o módulo dbcom. |
| dbcom_check | Verifica a situação do módulo dbcom. |
| suntracker_start | Inicia o módulo suntracker. |
| suntracker_stop | Para o módulo suntracker. |
| suntracker_check | Verifica a situação do módulo suntracker. |
| tcpcom_start | Inicia o módulo tcpcom. |
| tcpcom_stop | Para o módulo tcpcom. |
| tcpcom_check | Verifica a situação do módulo tcpcom. |
| tcpcom_setzero | Define as variáveis de posição do sistema para o valor zero. |
| tcpcom_timetracking | Define o modo de operação para posição a partir localização e horário. |
| tcpcom_sensortracking | Define o modo de operação para posição a partir do valor dos sensores. |
| tcpcom_manualtracking | Define o modo de operação para posição manual. |
| tcpcom_setsensors_[número]_[número]_[número]_[número] | Define valor para os sensores (para testes). |
| tcpcom_setpos_[número]_[número] | Define valores para a posição atual da placa. |
| tcpcom_setdesiredpos_[número]_[número] | Define valores para a posição desejada da placa. |
| tcpcom_checkvariables | Verifica o valor das variáveis do sistema. |
| uartcom_start | Inicia o módulo uartcom. |
| uartcom_stop | Para o módulo uartcom. |
| uartcom_check | Verifica a situação do módulo uartcom. |

**Fonte: Autoria Própria.**

### 3.7.4.2 Módulo UARTcom.py

Como foi mencionado anteriormente, tanto o movimento dos motores, como a captura dos valores dos sensores de luminosidade serão feitos pela placa MICROPIC_DAELT. A comunicação entre o Raspberry PI e o PIC será através de comunicação UART.

O módulo UARTcom.py tem como função principal enviar via serial a quantidade de pulsos a ser incrementada ou decrementada de cada motor, e em

seguida esperar a resposta do PIC confirmando o movimento e os valores dos sensores de luminosidade.

A *string* a ser enviada deve ser no formato: “+XXX,-YYY$”, em que XXX é um número inteiro de pulsos para o movimento do motor 1 e YYY é um número inteiro de pulsos para o motor 2.Os valores são então recebidos pelo módulo UARTcom.py e as variáveis centrais do módulo Main.py são então atualizadas. As mensagens enviadas pelo PIC via serial podem ser vistas na Tabela 7.

Os valores de pulsos movimentados são então multiplicados por uma constante de proporcionalidade, referente ao número de pulsos por ângulo, e somados com o valor de posição atual do sistema para que seja obtido o novo valor de posição. Tais constantes de proporcionalidade foram obtidas enviando solicitações de número de pulsos para os motores e verificando quantos graus foram movimentados. As constantes, obtidas de forma prática, são:

Motor 1 (Vertical): 1,87.

Motor 2 (Horizontal): 2,04.

#### 3.7.4.3 Módulo DBcom.py

Este é o módulo que tem a função de constantemente inserir no banco de dados os valores atuais de posição de ambos os motores, potência da placa, modo de operação e horário.

#### 3.7.4.4 Módulo TCPcom.py

Como o nome sugere, o módulo TCPcom tem como função principal estabelecer comunicação via TCP com interfaces externas. Assim que o módulo entra em execução ele se registra para receber mensagens, e um *socket* é criado esperando conexão na porta 12001.

Qualquer mensagem enviada para esta porta é executada pela função *cmdhandler* do módulo Main.py e funciona exatamente da mesma maneira como explicado no subtópico 3.7.4.1. Por exemplo, caso seja recebida a mensagem "DBcom_stop", o módulo DBcom.py é parado.

Outras funções úteis para o monitoramento e administração do sistema estão localizadas neste módulo, como por exemplo a função *setmanualmode*, que define o

sistema para o modo de posição manual. A lista com todas as funções deste e dos outros módulos pode ser encontrada em APENDICE F.

Na Figura 30, foi utilizado o aplicativo *TCPClient* (disponível para Android) para enviar o comando *tcpcom_checkvariables*, que verifica e informa o valor das principais variáveis do sistema.

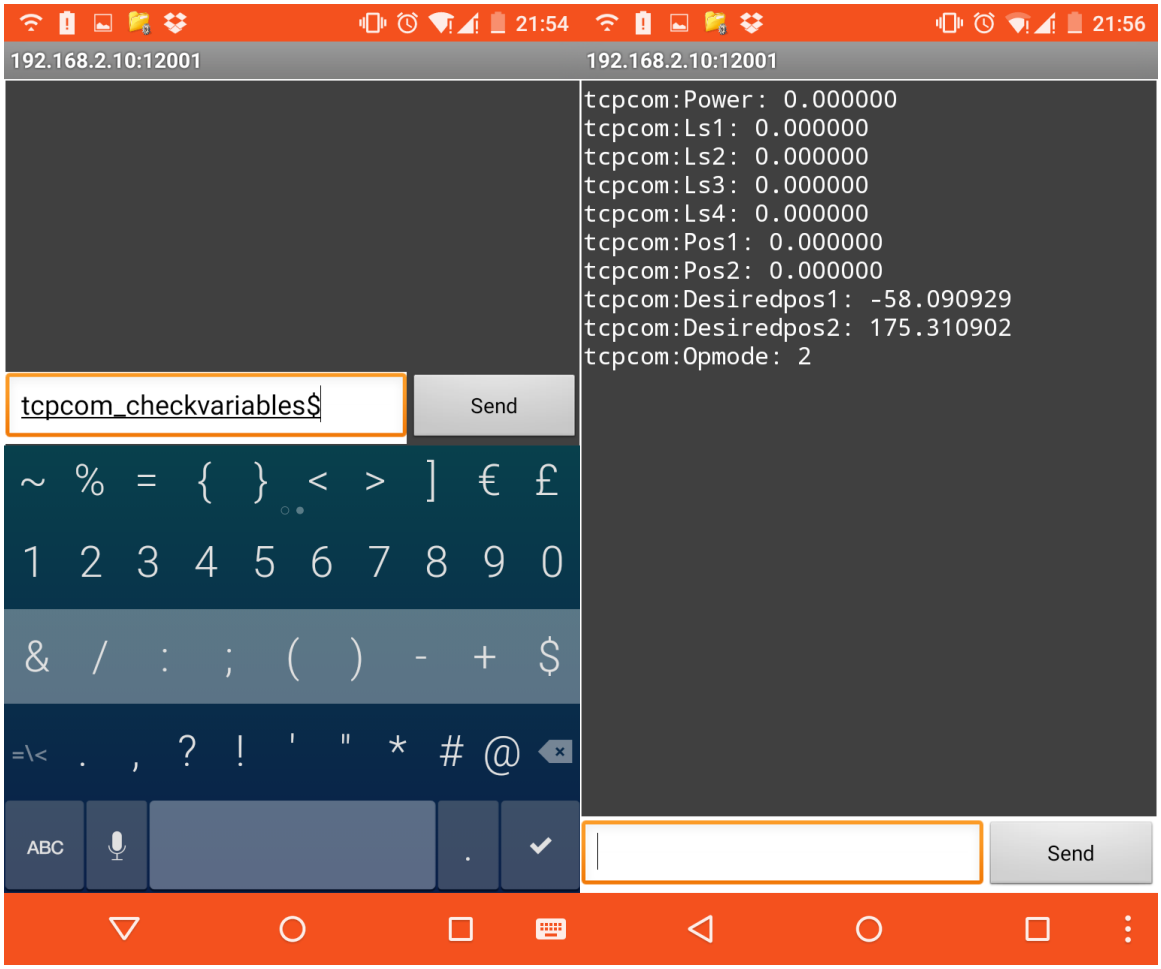



**Figura 30: Envio e recebimento de informações via TCP.**

**Fonte: Autoria Própria.**

### 3.7.4.5 Módulo suntracker.py

O módulo suntracker.py é o módulo responsável por verificar o módulo de operação do sistema e modificar as variáveis *desiredpos1* e *desiredpos2* de acordo com o tipo de lógica selecionada. Quando o modo de operação for 0 (zero), nenhuma alteração das variáveis descritas podem ser realizadas. Quando o modo de operação for 1 ou 2, o modo automático é ativado, permitindo que as variáveis sejam alteradas. No modo de operação 1, a nova posição é obtida através da diferença de tensão entre

os sensores de luminosidade. Já no modo de operação 2, a nova posição é obtida através do cálculo da posição do sol.

Para a programação do controle utilizam-se os 4 sensores para executar a lógica tanto para o motor 1 como para o motor 2.

Como mostra a Figura 16, para o motor da base os valores analógicos dos sensores do lado esquerdo da estrutura são somados (ls1 e ls3) e subtraídos os valores analógicos dos sensores do lado direito da estrutura (ls2 e ls4), como exposto na Equação 26.

$$Result_{azimute} = ls1 + ls3 - ls2 - ls4 \quad (26)$$

O valor de $Result_{azimute}$ é então multiplicada por uma constante ($cte$) que determinará a largura do passo que o dispositivo será submetido. Esse passo é então somado com a posição atual do dispositivo (chamada de $pos1$) e enviado para executar o movimento da posição desejada, chamada $d_{pos1}$, demonstrado na Equação 27.

$$d_{pos1} = pos1 + (Result_{azimute} * cte) \quad (27)$$

Para o motor de elevação é somado os valores analógicos dos sensores do lado superior da estrutura (ls1 e ls2) e subtraído os valores analógicos dos sensores do lado inferior da estrutura (ls2 e ls4), como mostrado na Equação 28.

$$Result_{elevação} = ls1 + ls2 - ls3 - ls4 \quad (28)$$

O valor de $Result_{elevação}$ é então multiplicada por uma constante ($cte$) que determinará a largura do passo que o dispositivo será submetido, esse passo então é somado com a posição atual do dispositivo (chamada de $pos2$) e enviado para executar o movimento da posição desejada, chamada $d_{pos2}$, visualizado na Equação 29.

$$d_{pos2} = pos2 + (Result_{elevação} * cte) \quad (29)$$

Enquanto que no modo 2, a posição é proveniente de uma modelagem que prevê a posição do sol a partir da data, hora e coordenadas geográficas, como visto no tópico 2.3.

A Figura 31 ilustra fluxo de dados entre os elementos do sistema.

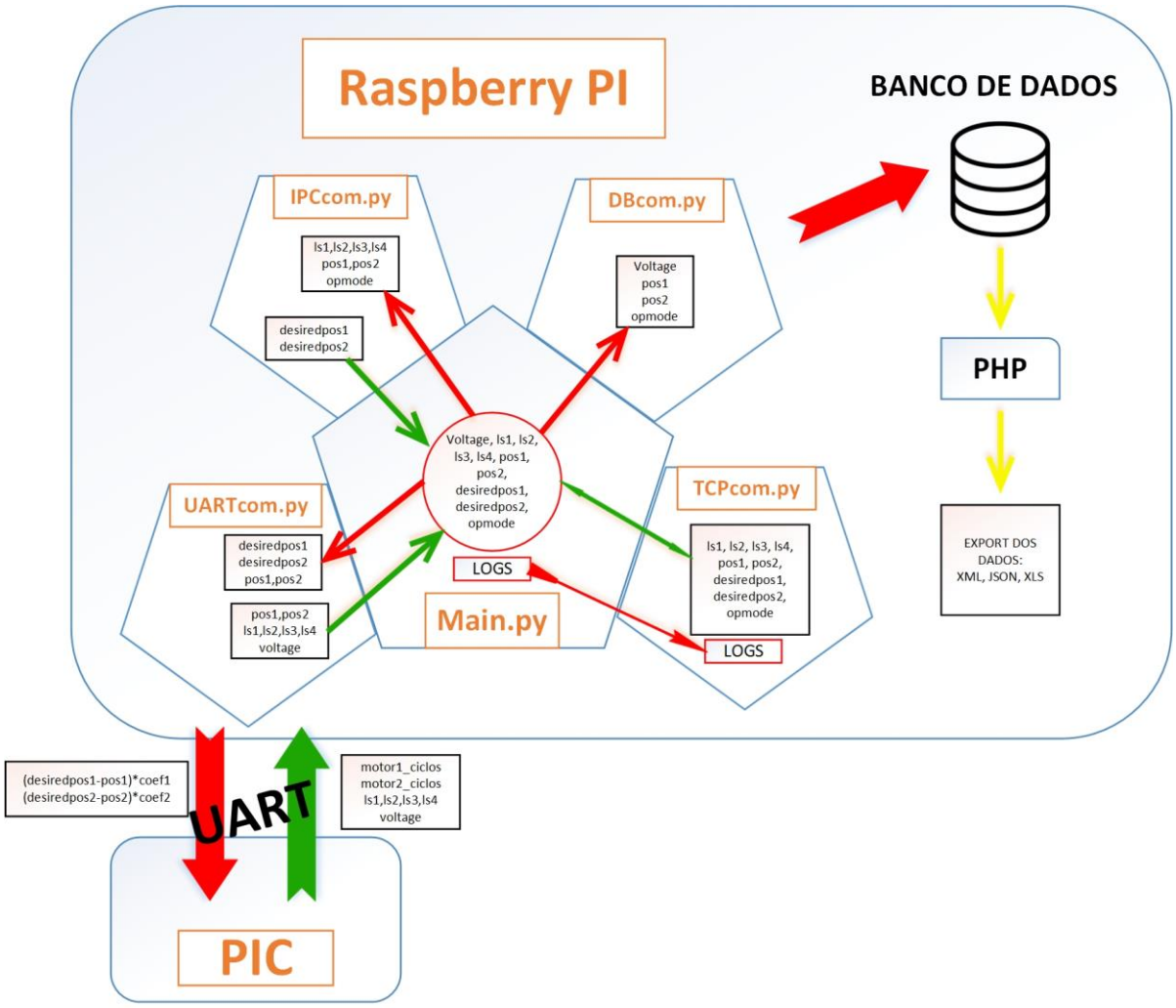



**Figura 31: Fluxo de dados do sistema.**

**Fonte: Autoria Própria.**

## 3.8 ESTRUTURA FÍSICA

A estrutura foi construída com o propósito de suportar a placa, de modo que possibilite que a mesma se movimente nos dois eixos polares estabelecidos. Além disso, foi projetada para que se necessite da menor força possível dos motores utilizados na movimentação de cada eixo, tornando o projeto menos custoso, e mais interessante do ponto de vista do retorno do investimento.

Na Figura 32 é apresentada uma visão geral da estrutura do rastreador (esboço), bem como os principais componentes, materiais e peças, utilizados para sua construção.

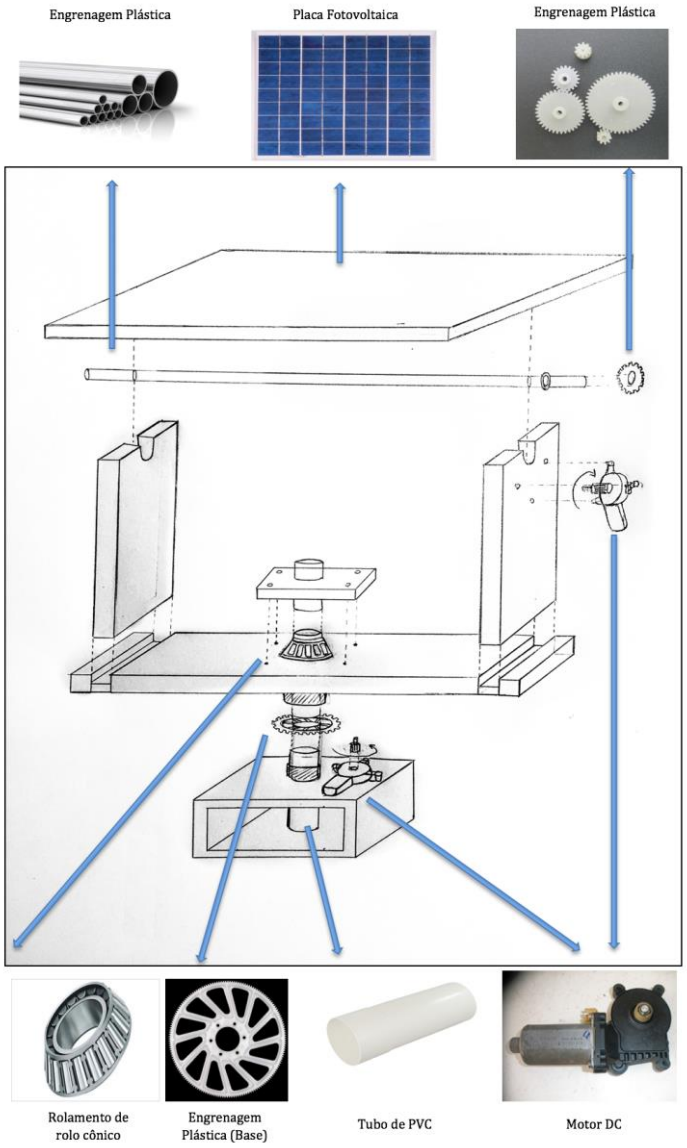



**Figura 32: Esquemático da Estrutura e Peças.**

**Fonte: Autoria Própria.**

Na Figura 33 é apresentada uma foto do sistema desenvolvido, composto pelo painel fotovoltaico e estrutura mecânica para posicionamento.

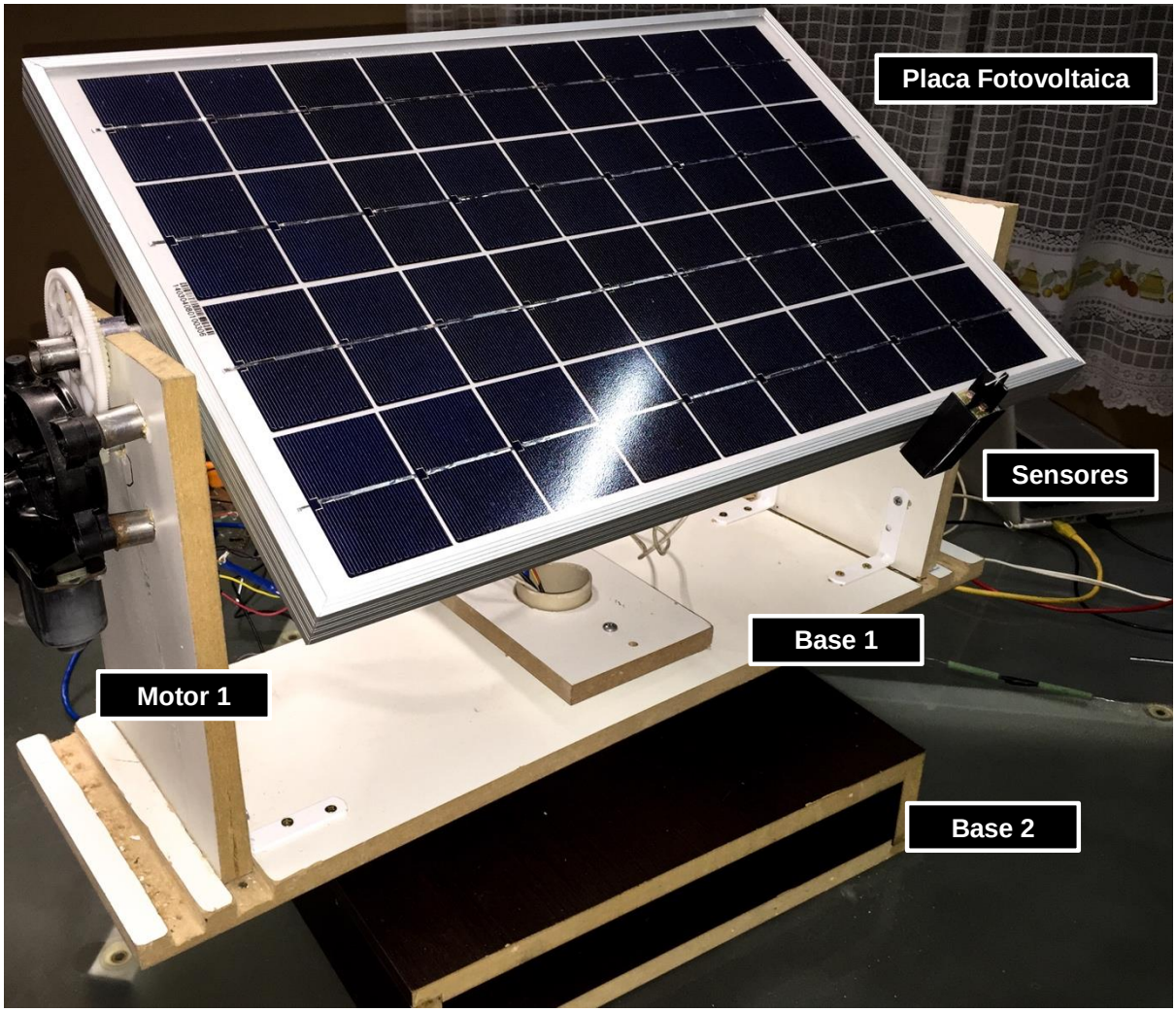



**Figura 33: Ilustração do sistema desenvolvido (vista Inclinada superior).**
**Fonte: Autoria Própria.**

A sustentação da estrutura se dá através da utilização de madeira MDF que pudesse ser trabalhada de forma facilitada, possibilitando adaptar o mesmo para as necessidades do projeto. De forma simplificada, foram usadas quatro peças MDF, sendo duas no contato direto com a placa que também oferece suporte ao motor do ângulo de elevação; uma utilizada de base para as primeiras; e uma última como base do sistema de rotação do motor de ângulo azimute (base).

Na Figura 34 é apresentada a visão laterial do sistema para melhor visualização da estrutura de giro da base.

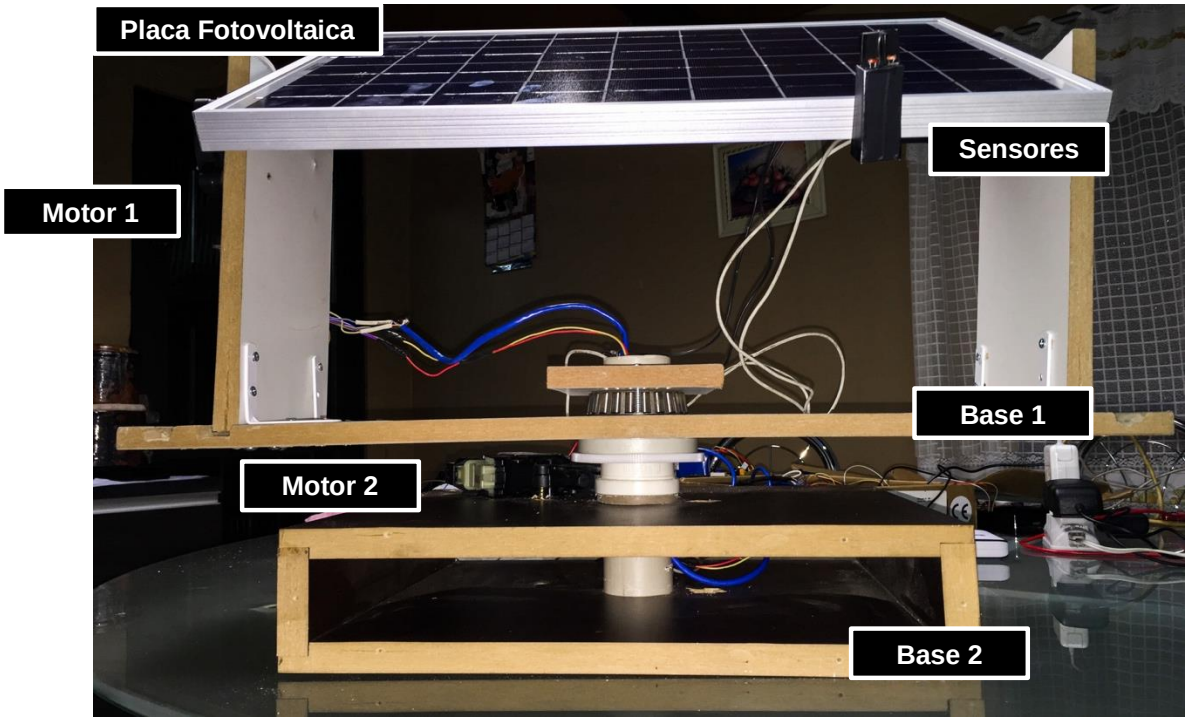



**Figura 34: Ilustração do sistema desenvolvido (vista frontal).**
**Fonte: Autoria Própria.**

A placa foi fixada em uma tubulação de ferro fundido (Figura 35), através de duas fitas *Hallermann* que passam por orifícios da lateral da placa fotovoltaica, de forma impeça sua rotação independente. O tubo metálico, por sua vez, é colocado sobre as laminas de madeira através de encaixes próprios para sua fixação na estrutura, que possibilitam uma estabilidade a placa mas ao mesmo tempo liberdade de rotação do eixo polar α, vertical.

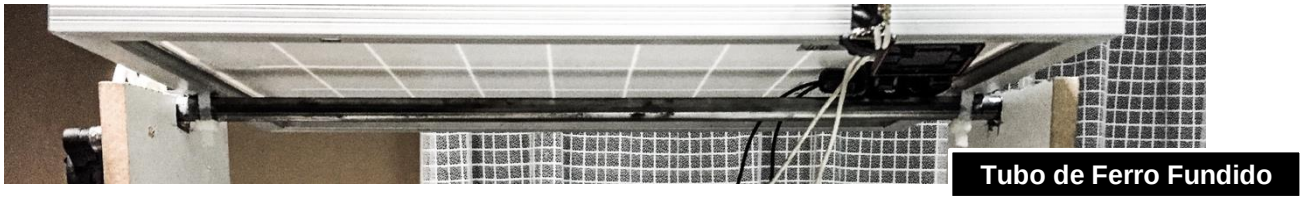



**Figura 35: Tubo de Ferro.**
**Fonte: Autoria Própria.**

A rotação no eixo polar α se dá através do motor 1 fixado em uma das placas de madeira verticais para que esse entre em contato com a engrenagem fixa na tubulação da placa, como mostrado nas Figuras 36 (visão frontal) e 37 (visão lateral).

A medida que esse motor é ligado o mesmo rotaciona a engrenagem, que por sua vez rotaciona a placa para o ângulo desejado.

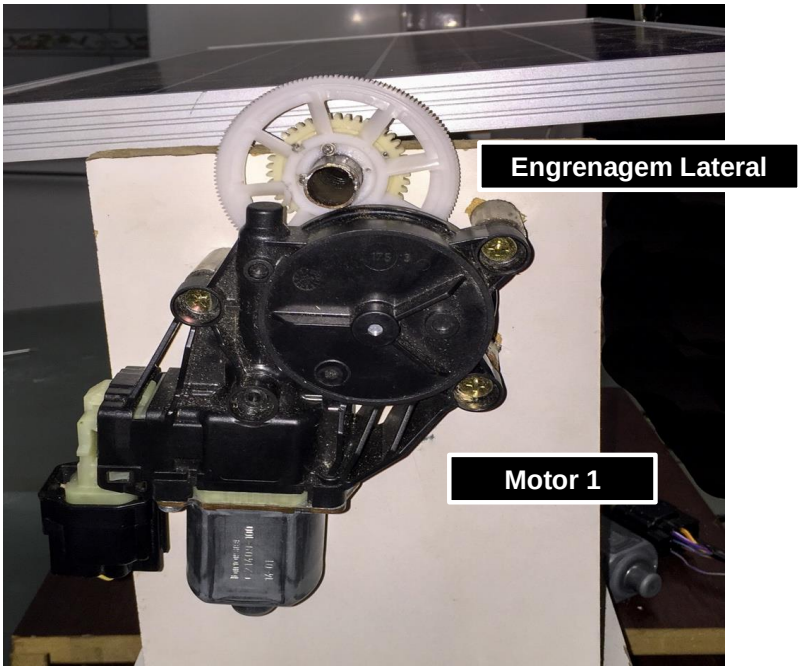



**Figura 36: Motor 1 de ângulo de elevação (vista frontal).**

**Fonte: Autoria Própria.**

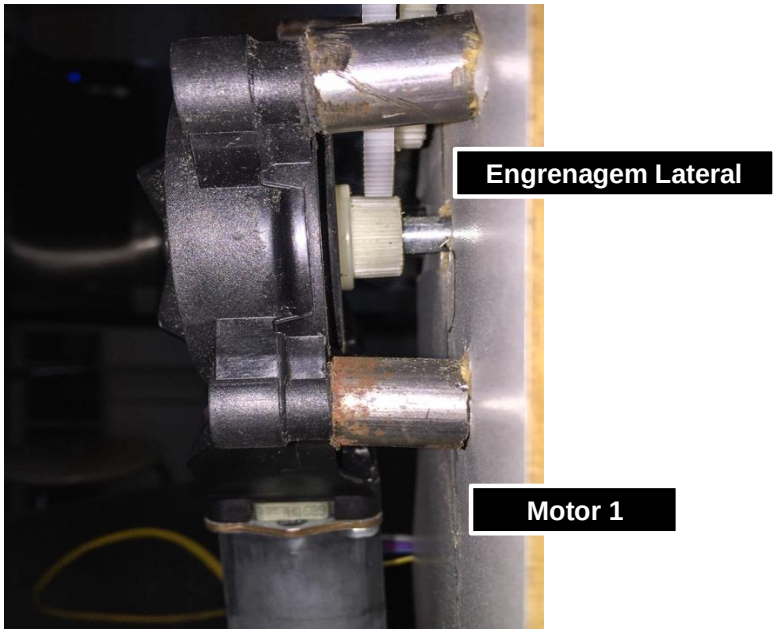



**Figura 37: Motor 1 de ângulo de elevação (vista lateral).**

**Fonte: Autoria Própria.**

A base horizontal é transpassada por uma tubulação de PVC (*Polyvinyl Chloride*) que é fixada na base do motor 2. Essa tubulação permite a ligação entre as

bases. Um rolamento de rolo cônico é usado na parte superior da base 1, que sustenta a placa, atribuindo uma liberdade de rotação no eixo polar γ, horizontal. Um suporte de madeira é usado sobre o rolamento para que esse se mantenha fixo na base 1, conforme mostrado na Figura 38.

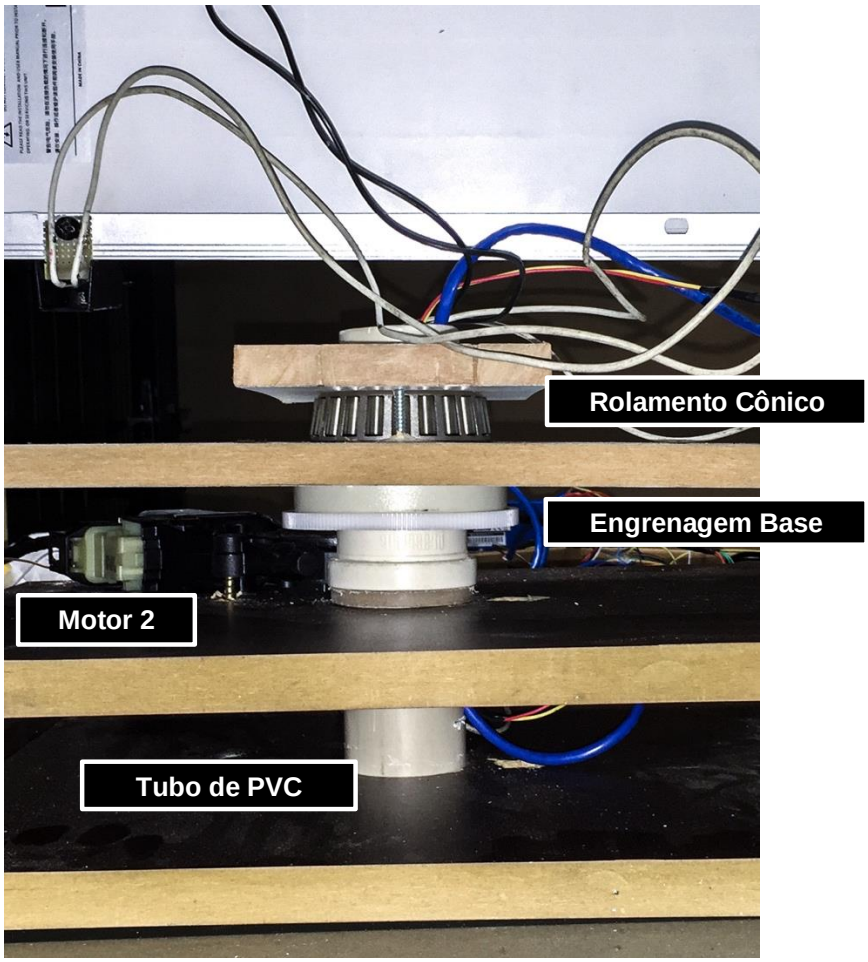



**Figura 38: Base da Estrutura.**
**Fonte: Autoria Própria.**

Na parte inferior da estrutura (Figura 39), entre as bases, tem-se a engrenagem plástica que é fixada na base 1, possibilitando que o motor 2 tenha ação sobre a estrutura da base 1 que sustenta a placa, permitindo a sua rotação horizontal.

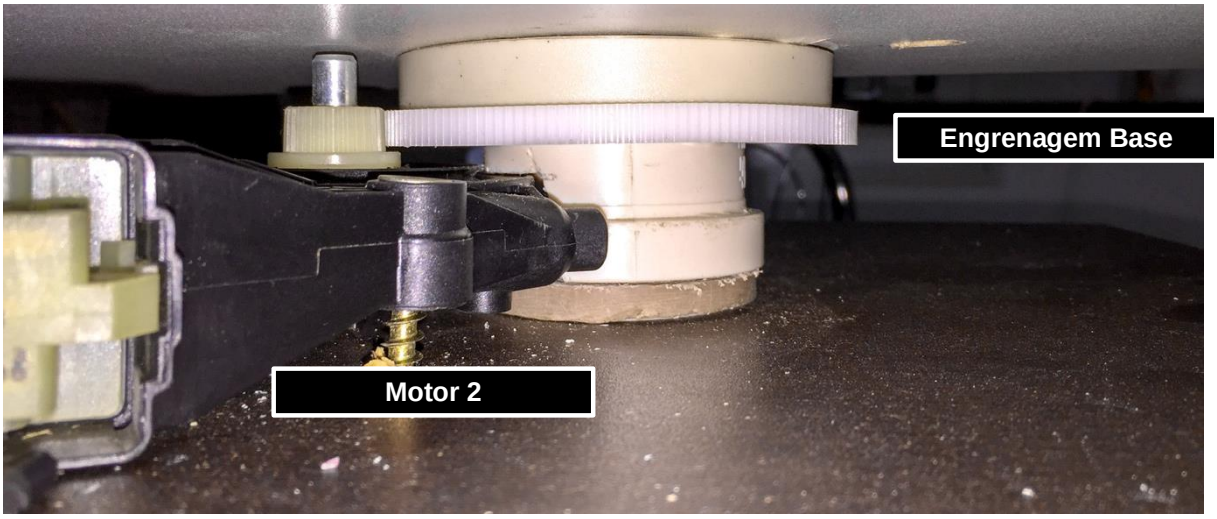



**Figura 39: Motor de ângulo azimute.**

**Fonte: Autoria Própria.**

A tubulação de PVC possibilita a passagem da fiação do projeto (Figura 40), tornando possível a rotação no eixo γ de forma mais eficaz. Deste modo, as conexões entre a placa fotovoltaica, o sensor e o motor 1 com a placa do PIC passam por essa passagem, que tem como ponto final a base 2, ficando fixa durante o processo de rotação.

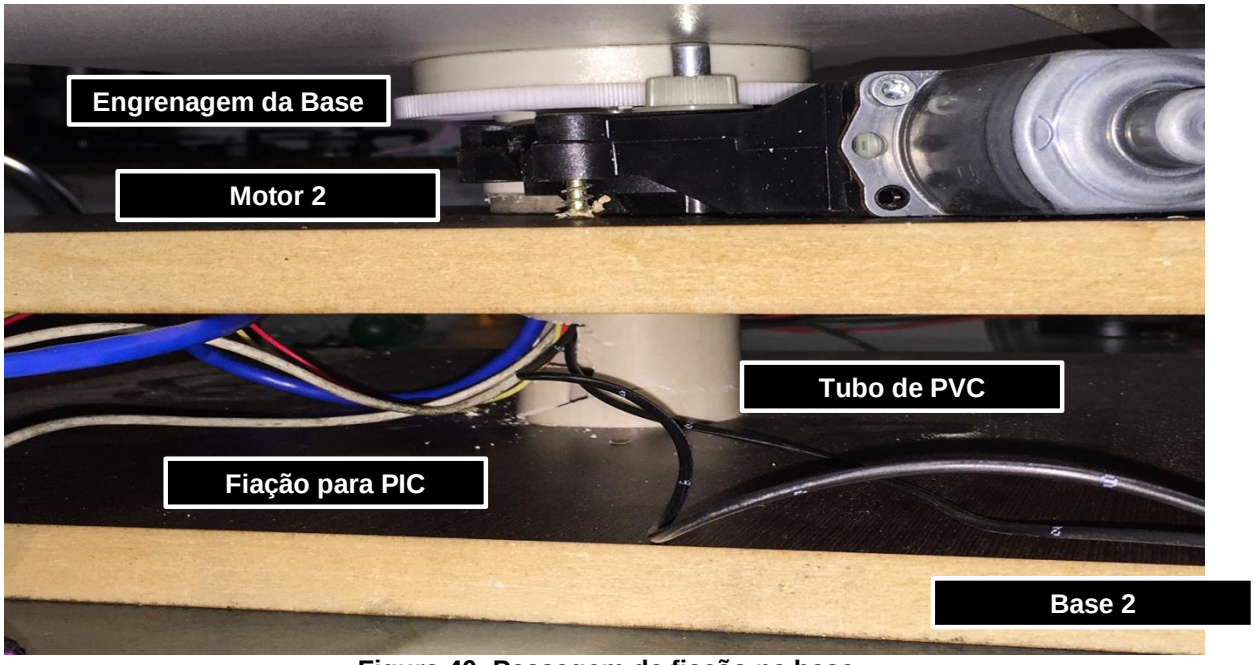



**Figura 40: Passagem da fiação na base.**

**Fonte: Autoria Própria.**

# 4. RESULTADOS

Após toda a configuração de *Software* e *Hardware*, foram realizados testes para verificar o funcionamento dos três modos de operação - manual, automático por sensores de luminosidade e automático por modelo de posicionamento solar -, dos três formatos de saídas de dados (JSON, XML e CSV), e da obtenção dos valores de tensão, corrente e potência.

## 4.1 MODO MANUAL DE OPERAÇÃO

Para o modo manual de operação, foram executados comandos de solicitação de posição utilizando o aplicativo de comunicação TCP, TCPConsole, como mostra a Figura 41. As posições solicitadas foram 15 e 0 (valores amostrais), para os ângulos vertical e horizontal, respectivamente.

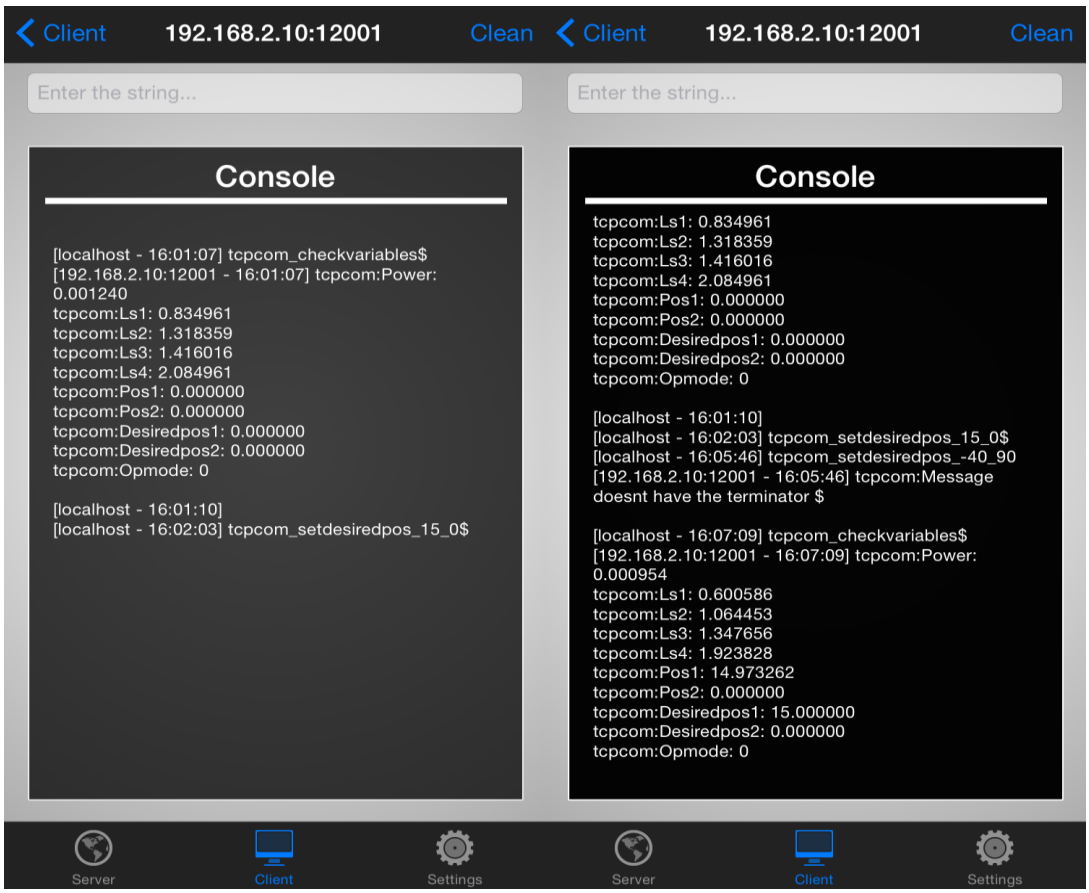



**Figura 41: Comandos através do aplicativo de comunicação TCP.**

**Fonte: Autoria Própria.**

Como a placa estava originalmente na posição 0 e 0, o motor horizontal não executou nenhum movimento. O movimento vertical para 15° pode ser visto na Figura 42. A Figura 43 mostra a medida do ângulo de inclinação após o movimento.

**Figura 42: Movimento manual da posição vertical: antes (foto à esquerda) e depois (foto à direita).**

**Fonte: Autoria Própria.**

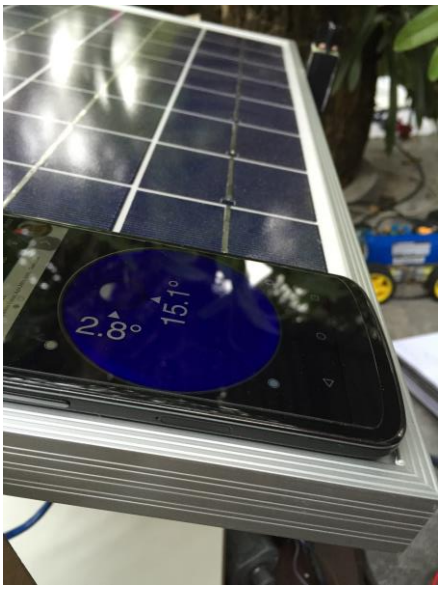



**Figura 43: Movimentação manual para a posição de 15°.**

**Fonte: Autoria Própria.**

Para o segundo movimento testado, os ângulos escolhidos foram -40 (vertical) e 90 (horizontal). A Figura 44 mostra a solicitação de mudança de posição e as variáveis do sistema após a mudança.

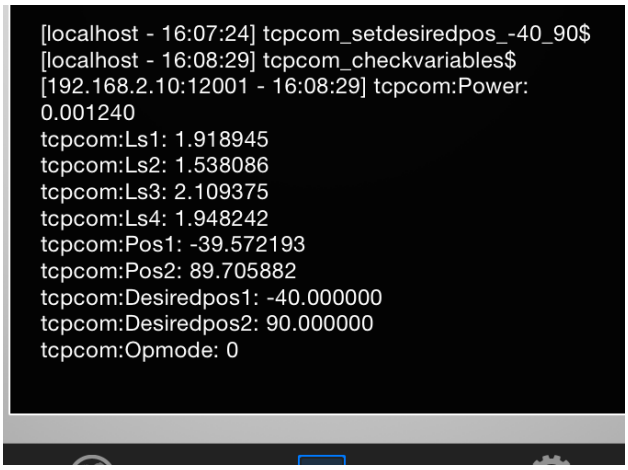



**Figura 44: Envio de comando para movimentação manual.**

**Fonte: Autoria Própria.**

Na Figura 45, pode ser visualizado o movimento do motor horizontal, que rotacionou a placa em 90 graus negativos, como solicitado.

**Figura 45: Posição da placa após movimentação horizontal de 90°.**

**Fonte: Autoria Própria.**

## 4.2 MODO AUTOMÁTICO DE OPERAÇÃO POR SENSORES

Como os testes foram feitos em dia com muitas núvens e pouca presença de luminosidade, foi utilizada uma lâmpada para testar o posicionamento da placa solar a partir da medida dos sensores. A Figura 46 mostra as posições inicial e final da estrutura.

**Figura 46: Movimentação automática via sensores.**
**Fonte: Autoria Própria.**

Os dados do período em questão foram obtidos a partir de conexão com o Web Server, utilizando os seguintes links:

- JSON: http://192.168.2.10/index.php?mode=json&select=pos1,pos2,ls1,ls2,ls3,ls4,opmode,time&opmode=1&btime=17:00:00&etime=17:15:00
- XML: http://192.168.2.10/index.php?mode=xml&select=pos1,pos2,ls1,ls2,ls3,ls4,opmode,time&opmode=1&btime=17:00:00&etime=17:15:00
- CSV: http://192.168.2.10/index.php?mode=excel&select=pos1,pos2,ls1,ls2,ls3,ls4,opmode,time&opmode=1&btime=17:00:00&etime=17:15:00

Observa-se que ao longo dos cerca de 3 minutos que levou para a placa atingir a estabilidade, os valores dos sensores se tornaram cada vez mais próximos, evidênciando o fato da placa estar voltada à luz. A Figura 47 mostra a posição dos motores de movimento horizontal (motor 1) e vertical (motor 2) ao longo do tempo.

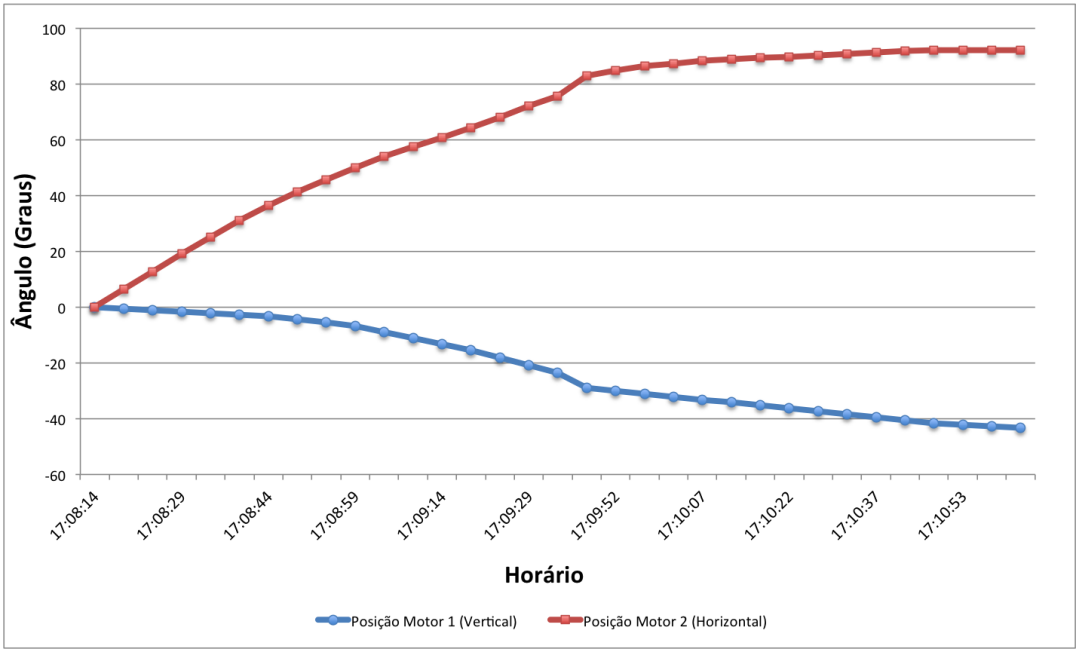



**Figura 47: Posições dos motores.**

**Fonte: Autoria Própria.**

## 4.3 MODO AUTOMÁTICO DE OPERAÇÃO PELO CÁLCULO DA POSIÇÃO DO SOL

O modo de operação 2 é o modo automático que se baseia em modelo da posição do sol. A Figura 49, mostra a solicitação do posicionamento através do modo 2, e a obtenção das variáveis do sistema após o movimento. O modelo de posição solicitou os valores de ângulo -64,65° e 134,80°, para elevação e azimute, respectivamente. A Figura 50 mostra a localização do sol a partir do aplicativo *Sun Position Calculator*, que indica 312,62° (Azimute) e 23,21° (elevação).

A diferença dos valores encontrados é pela convenção adotada para o dispositivo. Na Figura 48 é demonstrada a convenção para o dispositivo correspondente ao ângulo azimute (B), sendo de 0° até 180° crescente de forma continua. Enquanto, que para os cálculos adotados no aplicativo *Sun Position Calculator* no período da manhã, se inicia com cerca de 90° diminuindo até 0° e no período da tarde se inicia com 360° diminuindo até cerca de 270° (*PVEducation*, 2015).

O ângulo de azimute 312,62°, dado pelo aplicativo representa para o sistema cerca de 137,38° e o ângulo calculado pelo sistema foi de 134,80°. A diferença de valores no ângulo de elevação é devido à posição inicial da placa ser horizontal, como visto na Figura 46 (lado esquerdo). Assim, é necessário a diminuição de 90° do ângulo calculado. O ângulo de elevação -64,65° apresentado na saída do dispositivo corresponde ao ângulo real calculado de 25,35°, menos os 90°.

A diferença de cerca de 2° apresentado em ambos os ângulos é devido a pequena diferença do horário inserido para efetuar-se o cálculo da posição pelo aplicativo.

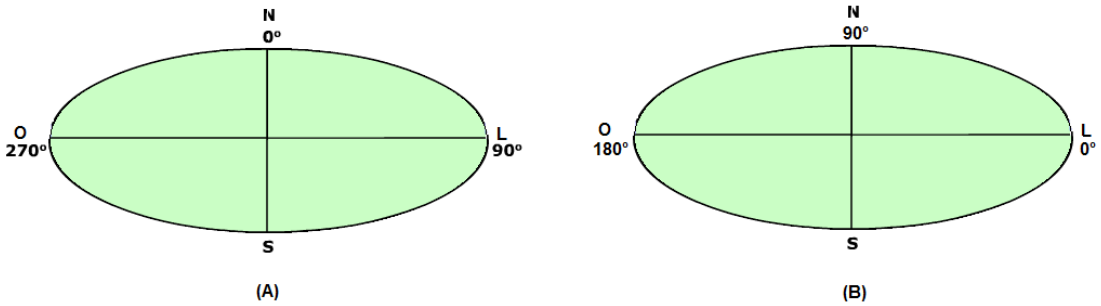



**Figura 48: (A) ângulo de azimute pelo *Sun Position Calculator* (B) ângulo de azimute pelo sistema.**

**Fonte: Autoria Própria.**

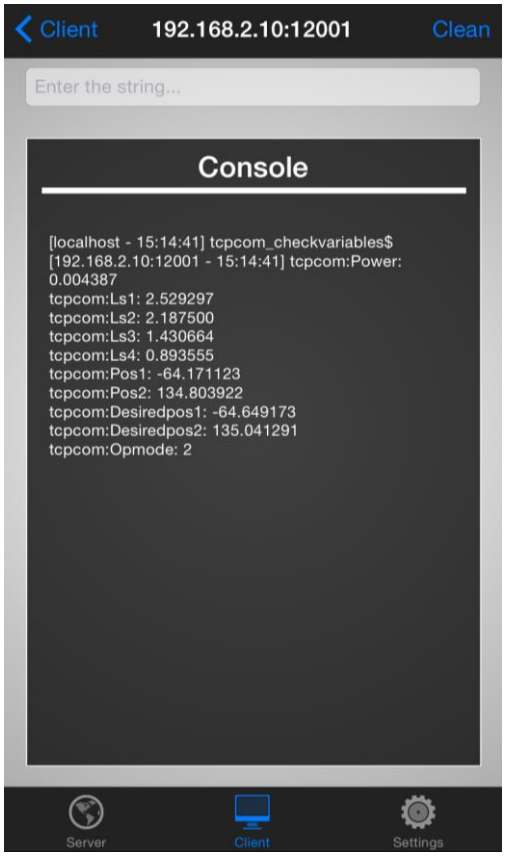



**Figura 49: Aplicativo TCPConsole.**

**Fonte: Autoria Própria.**

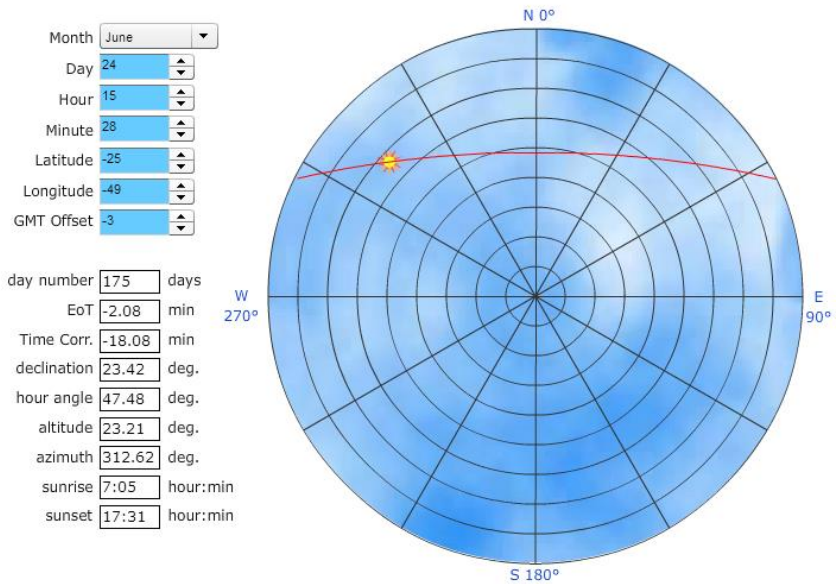



**Figura 50: Posição do Sol.**

**Fonte: PVEducation (2015).**

Como mostra a Figura 51, a placa realmente alcançou a inclinação solicitada pelo programa.

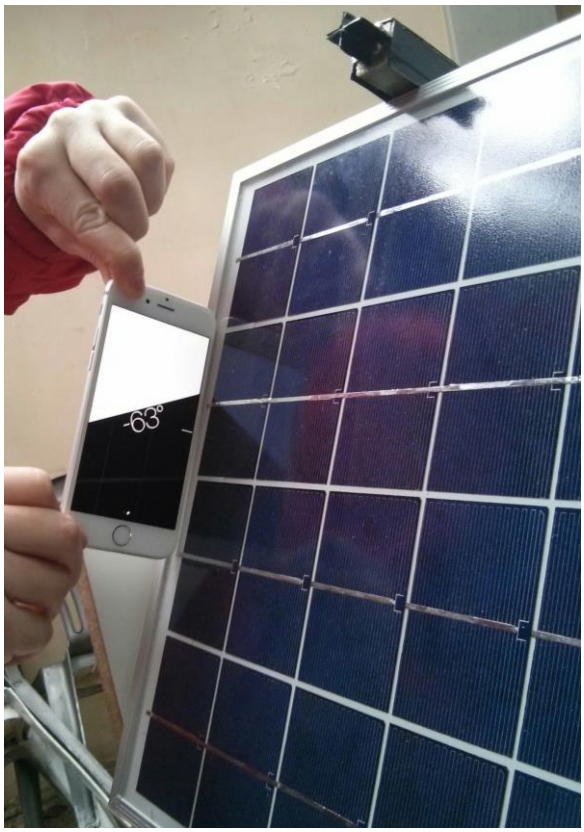



**Figura 51: Inclinação da placa.**

**Fonte: Autoria Própria.**

A Figura 52 e a Figura 53 mostram os valores de posição dos motores em função do tempo decorrido. Observa-se que tanto para o motor 1 como para o motor 2 a movimentação ocorre em pulsos, comprovando a lógica da contagem de pulsos apresentada anteriormente.

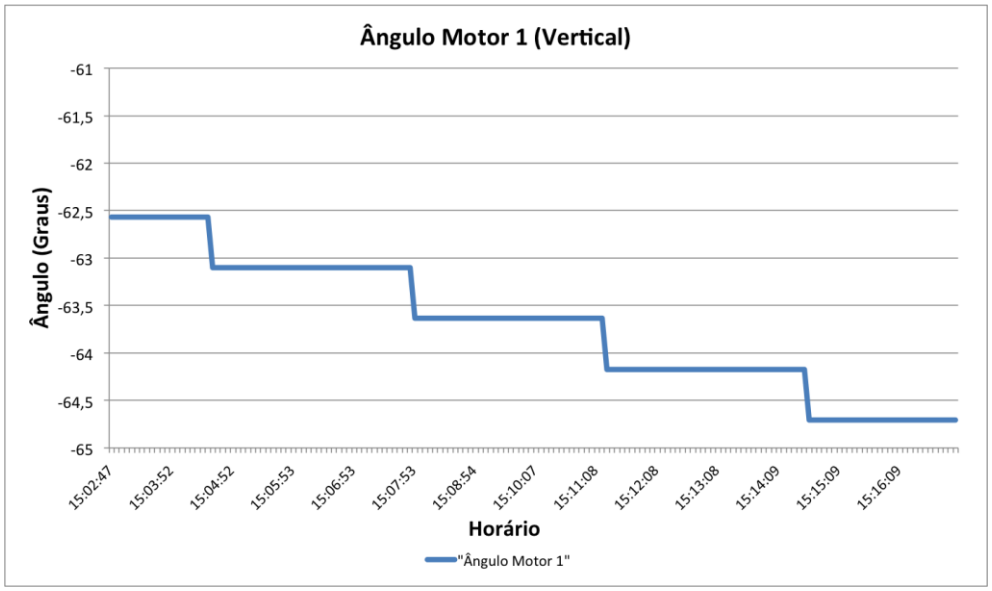



**Figura 52: Posição por Tempo, Motor 1.**

**Fonte: Autoria Própria.**

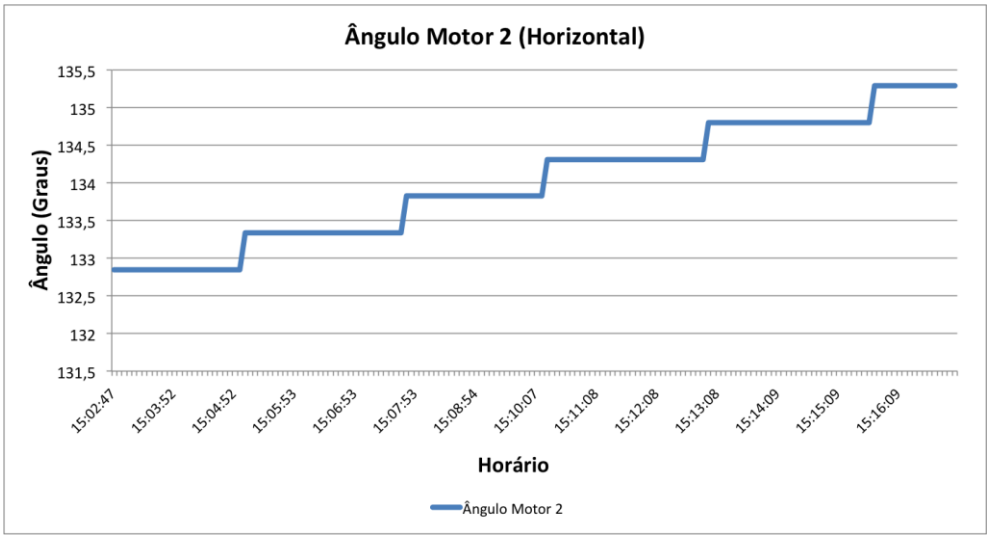



**Figura 53: Posição por Tempo, Motor 2.**

**Fonte: Autoria Própria.**

## 4.4 POTÊNCIA, TENSÃO E CORRENTE

A medições foram realizadas utilizando o modo de operação manual, colocando os dois motores na posição de 0°. Foram configuradas as capturas para intervalos de trinta segundos, o que possibilitou que fossem obtidas até mesmo pequenas nuancias na luminosidade. Ao longo de um dia também foram feitas medidas por um multímetro, para comprovar a veracidade dos dados coletados pela placa. A Figura 54 mostra os valores de tensão, corrente e potência obtidos das 10 h às 18 h pelo sistema. Foi incluído uma curva de tendência polinomial de grau 5, para facilitar a análise dos dados obtidos.

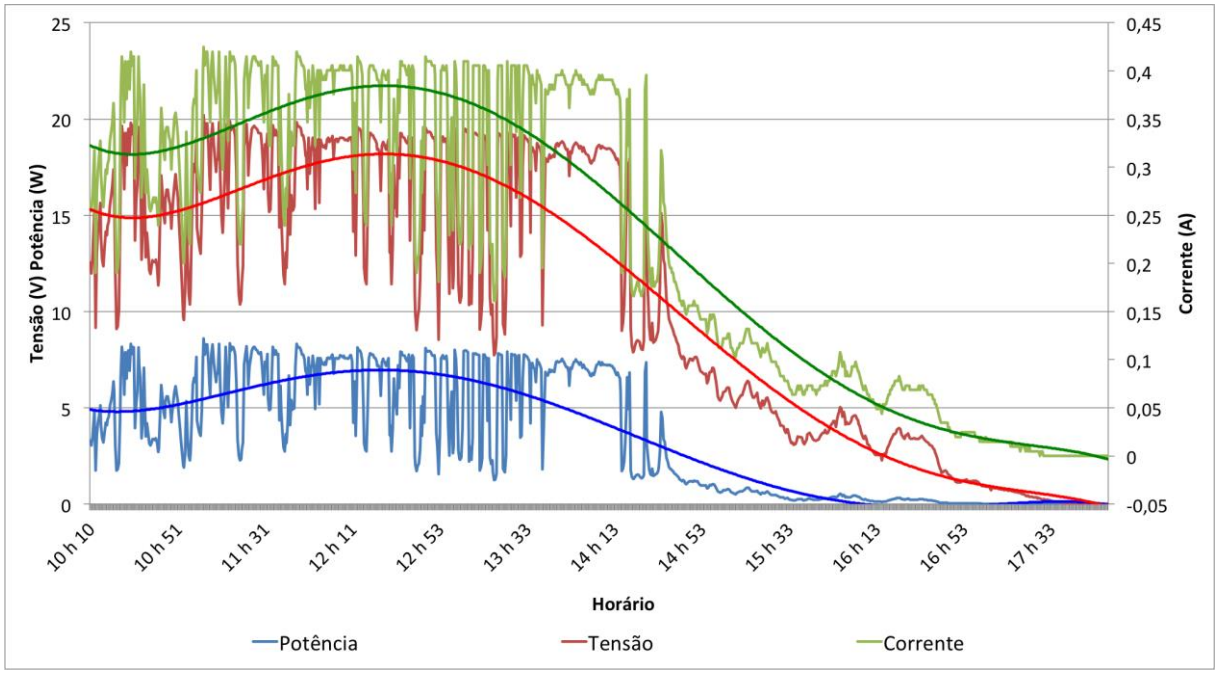



**Figura 54: Medições de tensão, corrente e potência realizadas pelo sistema**

**Fonte: Autoria Própria.**

Na Figura 55 é ilustrado o gráfico com as medições realizadas manualmente através de um multímetro digital. Pode-se constatar a similaridade entre as curvas dos gráficos da Figuras 54 e 55, além que pontualmente os dados coincidem, indicando a confiabilidade na obtenção dos valores pelo sistema.

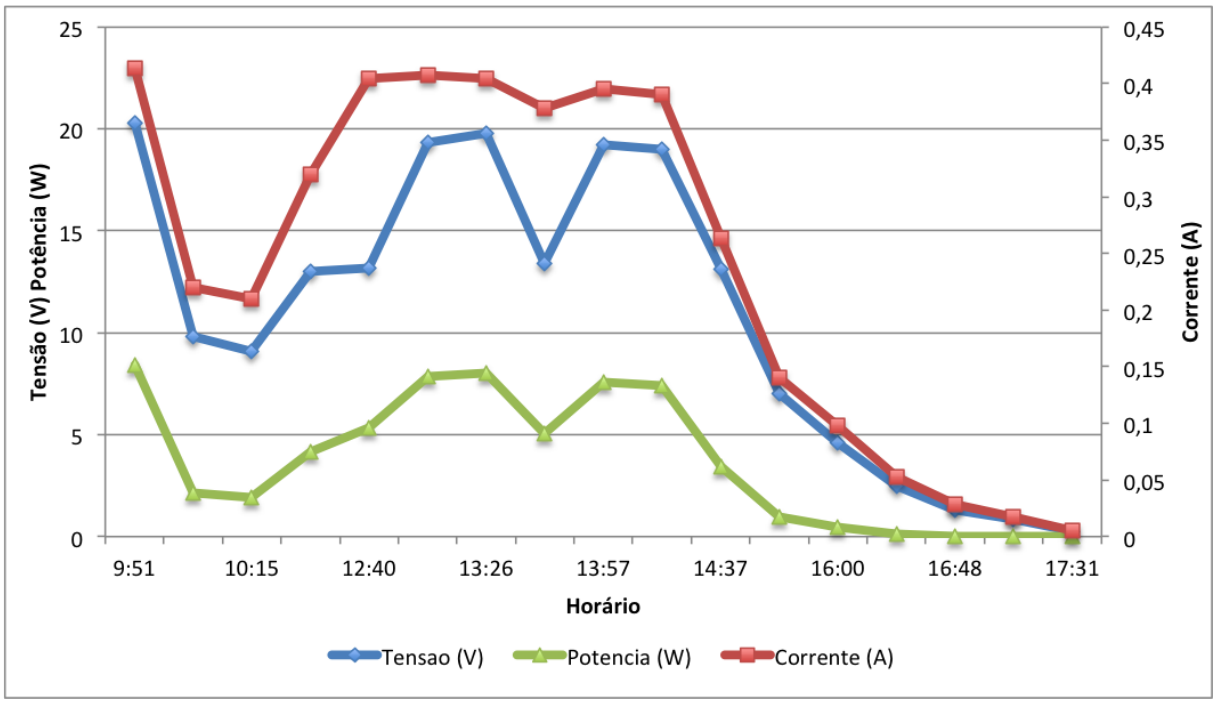



**Figura 55: Medições de tensão, corrente e potência realizadas manualmente**

**Fonte: Autoria Própria.**

## 5. DISCUSSÃO E CONCLUSÕES

A otimização da matriz energética do planeta como pensamento futuro passa pela otimização do aproveitamento das fontes de energias renováveis limpas, como a energia solar, abundante e de fácil obtenção. Em virtude da escassez dos recursos energéticos evidenciado nas últimas décadas, o incentivo para o desenvolvimento tecnológico visando melhor aproveitamento tende fomentar novas tecnologias a partir de mecanismos com melhor eficiência e economicamente mais viáveis.

O projeto procurou desenvolver um sistema mecânico simples que atendesse os requesitos necessários a sustentação e rotação da placa fotovoltaica, em dois graus de liberdade, através da utilização de motores de baixo custo e precisão adequada.

O dispositivo desenvolvido, fundamentado na rastreabilidade solar, tem capacidade para operar em diferentes regiões e possui diferentes modos de operação. Através de controle sensoriado, por trajetória calculada ou modo com ângulo fixo, a placa solar pode ser posicionada de forma a atender as necessidades do usuário final.

O projeto conta com a manipulação dos valores das variáveis do processo que são enviadas para o banco de dados desenvolvido no Raspberry PI e é disponibilizado para o usuário final para posterior análise, assim é constatada a integração entre os diferentes componentes e funcionalidades do projeto.

A utilização de um motor DC para movimentação da placa foi um ponto de dúvida quanto à precisão do sistema. Após testes utilizando os sensores *hall*, observou-se que, caso não ocorra derrapagem nas engrenagens, a precisão é aceitável. A calibração, entretanto, é recomendada de tempos em tempos para que os valores continuem confiáveis. Como uma sugestão de melhora, um sensor de posição poderia ser instalado na posição zero de cada um dos motores para zerar o valor de posição quando forem ativados.

Embora o Raspberry PI seja um equipamento com inúmeros recursos, desde poder de processamento poderoso, com saída para vídeo em alta resolução, áudio, vídeo e internet, foi necessária a implementação de outra placa microprocessada para alguns recursos necessários, como a obtenção de sinais analógicos e movimento dos motores.

É notável que há um pequeno atraso entre movimentos do motor quando o modo de operação é automático via sensores. O atraso se deve exatamente por que

os valores dos sensores de luminosidade são obtidos apenas após a série de movimento dos dois motores, e após isso a rotina de cálculo a partir da diferença entre o valor dos sensores pode ser executada novamente.

Observou-se que o sistema em módulos foi totalmente válido, principalmente para alterações e novas implementações, que foram se tornando mais simples. A execução do programa, embora envolva uma série de *threads*, não foi alta o suficiente para gerar impactos visíveis para o sistema como um todo.

## REFERÊNCIAS

Adafruit, 2015. **Raspberry PI B+ DataSheet**. Disponível em:<http://www.adafruit.com/product/2358>. Acesso em: 10 Mar. 2015.

ALDABÓ, R.. **Energia Solar**. São Paulo: Artliber Editora, 2002.

ANDRADE, G.; HOLANDA JR., F.; SYPNIEVSKI, R. **Controlador de carga microcontrolado para painel fotovoltaico com conexão paralela e regulagem de ponto de máxima potência**. 2006. TCC (Graduação em Engenharia Industrial Elétrica) - Departamento Acadêmico de Eletrotécnica, Universidade Tecnológica Federal do Paraná, Curitiba, 2006.

ANEEL, 2014. **Perspectivas da Energia Solar.** Disponível em: <http://www.aneel.gov.br/hotsite/mmgd/slides/Antonio%20Carlos%20de%20Andrada%20Tovar.pdf>. Acesso em: 28 dez. 2014.

Blue Point Engineering, 2007. **Servomotor Information**. Disponível em: <http://www.bpesolutions.com>. Acesso em: 28 dez. 2014.

CHONG, K., WONG, C., TUNKU, U. e RAHMEN. **General Formula for On-Axis Sun Tracking System.** Universiti Tunkun adbul Rahman Malaysia, 2014. Capitulo 13 pagina 266.

CODY, J. et al. **Regenerative Braking in a Electric Vehicle**. Disponível em: <http://www.komel.katowice.pl/ZRODLA/FULL/81/ref_20.pdf>. Acesso em 20 jun. 2015.

COLLE S. A.; PEREIRA E. B. **Atlas de Irradiação Solar do Brasil (Primeira Versão para Irradiação Global Derivada de Satélite e Validada na Superfície)**, Instituto Nacional de Meteorologia INMET, Brasília, 1998.

COOK, G., BILLMAN, L., ADCOCK, R. **Photovoltaic Fundamentals.** Department of Energy, 1995.

ELECTRICAL CONSTRUCTION & MAINTENANCE, 2014. **The Highs and Lows of Photovoltaic System Calculations**. Disponível em:<http://ecmweb.com/green-building/highs-and-lows-photovoltaic-system-calculations>. Acesso em: 10 Mar. 2015.

EMBEDDED LINUX WIKI, 2014. **RPi Hub**. Disponível em: <http://elinux.org/RPi_Hub>. Acesso em: 20 fev. 2015.

Energy Trend, 2014. **Photovoltaic Costs Could Fall 40 percent by 2015.** Disponível em: <http://pv.energytrend.com/research/Barber_PV_20120524.html>. Acesso em: 18 dez. 2014.

FRADEN, J., **Handbook of Modern Sensors:** Physics, Designs, and Applications. New York: Springer, 2010.

FRAIDENRAICH, N., VILELA, O.C., TIBA, C. **Photovoltaic pumping systems driven by tracking collectors. Experiments and simulation.** 2003.

GSMNATIONBLOG, 2013. **Raspberry Pi Model A Now Available to Public in Europe**. Disponível em: <http://www.gsmnation.com/blog/2013/02/05/raspberry-pi-model-a/>. Acesso em: 20 fev. 2015.

IMAGES SCIENTIFIC INSTRUMENTS, 2014. **Photovoltaic Cell – Generating eletricity.** Disponível em: <http://www.imagesco.com/articles/photovoltaic/photovoltaic-pg4.html>. Acesso em: 20 fev. 2015.

INEE, 2014. **Geração Distribuída.** Disponível em: <http://www.inee.org.br/forum_ger_distrib.asp>. Acesso em: 05 dez. 2014.

INTERWORKX, 2014. **InterWorx Clustering on Raspberry Pis!**. Disponível em: <http://www.interworx.com/community/interworx-clustering-on-raspberry-pis/>. Acesso em: 20 fev. 2015.

LUQUE, A.; HEGEDUS, S. **Handbook of photovoltaic science and engineering. Instituto de Energia Solar**, Universidade Politécnica de Madri. Madri: Wiley, 2. ed., 2011.

Micro Chip, 2015. **PIC16F877A**. Disponível em: <http://www.microchip.com>. Acesso em: 15 mar. 2015.

PINOUT. **The comprehensive Raspberry Pi Pinout guide for the Raspberry Pi**. Disponível em: <http://pi.gadgetoid.com/pinout>. Acesso em: 20 fev. 2015.

PORTAL ECOTECH, 2014. **Photovoltaic technology**. Disponível em: <http://www.1portal.net/Ecotech/html/solartech.html>. Acesso em: 20 fev. 2015.

PRINSLOO, G.; DOBSON, R. **Solar Tracking.** South Africa: Prinsloo & Dobson, 2014.

PVEducation, 2015. **Sun Position calculator.** Disponivel em: <http://www.pveducation.org/pvcdrom/properties-of-sunlight/sun-position-calculator>. Acesso em: 20 Mai. 2015.

RASPBERRY PI FOUNDATION, 2012. **Raspberry Pi Documentation**. Disponível em: <http://www.raspberrypi.org/documentation/>. Acesso em: 08 fev. 2015.

RASPI.TV, 2015 **Raspberry Pi Family Photo Update – The Red B+ and 256 Mb Rev 2 B**. Disponível em: <http://raspi.tv>. Acesso em: 20 fev. 2015.

RAWLINGS, J. B. **Turorial Overview of Model Predictive Control.** IEEE Control System Maganize, 2000.

SOLARGIS, 2014. **Map of Direct Normal Irradiation - Brasil.** Disponível em:<http://solargis.info/doc/_pics/freemaps/1000px/dni/SolarGIS-Solar-map-DNI-Brazil-en.png>. Acessado em: 15 Fev. 2015.

STEPHAN, R. M. **Acionamento, comando e Controle de Máquinas Elétricas.** Rio de Janeiro. Editora Ciência Moderna Ltda., 2013.

VILLALVA, M. G.; GAZOLI, J. R. **Energia Solar Fotovoltaica**: Conceitos e Aplicações. São Paulo: Érica, 2012.

WALT, K. R. **The data conversion handbook**. United States of America: Analog Devices Ink, 2014.

YINGLI SOLAR, 2014. **Manual do fabricante - Série de células YGE 72**. Disponível em: <http://www.yinglisolar.com/br/downloads/>. Acesso em: 20 fev. 2015.

## ANEXO A – DATASHEET MOTOR FPG 2

O manual do fabricante dos motores descreve informações e características de desempenho dos motores utilizados.

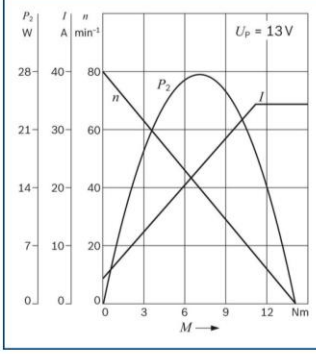
**Technical data**

| | | |
|---|---|---|
| Part number | | **0 130 822 492** |
| mirror-image | | **0 130 822 493** |
| Nominal voltage | $U_N$ | 12 V |
| Nominal power | $P_N$ | 14 W |
| Nominal current | $I_N$ | 10 A |
| Maximum current | $I_{max}$ | 34 A |
| Nominal speed | $n_N$ | 67 min$^{-1}$ |
| Nominal torque | $M_N$ | 2 Nm |
| Breakaway torque | $M_A$ | 13,5 Nm |
| Reduction | i | 73 : 1 |
| Direction of rotation | | L/R |
| Type of duty | | S 2 |
| Degree of protection | | IP 54 |
| Weight | | approx. 0,59 kg |

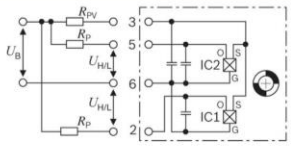
Clockwise: I on (+), II on (-)
Anti-clockwise: II on (+), I on (-)

Connector Housing:
Yazaki 7287-1116-30

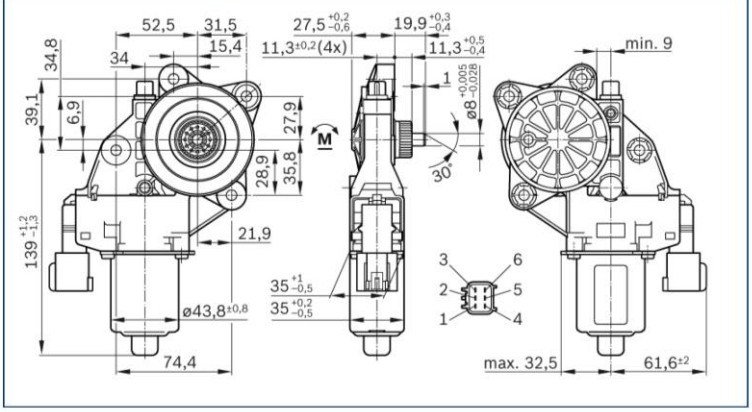
**Characteristic curve**




**Connection diagram**




**Dimensional drawing**




Robert Bosch GmbH
Automotive Aftermarket
Postfach 410960
76225 Karlsruhe
Germany

www.bosch-elektromotoren.de

BOSCH
Invented for life

## ANEXO B – MANUAL DO FABRICANTE DA PLACA FOTOVOLTAICA

O manual do fabricante mostra importantes informações e características da placa fotovoltaica que foi adquirida, em diferentes condições.

YGE 20 SERIES

YL020P-17b 1/6

YINGLI SOLAR

### ABOUT YINGLI GREEN ENERGY

- Yingli Green Energy Holding Company Limited (NYSE: YGE) is one of the world's largest fully vertically integrated PV manufacturers. With over 2 GW of modules installed globally, we are a leading solar energy company built upon proven product reliability and sustainable performance. We are the first renewable energy company and the first Chinese company to sponsor the FIFA World Cup™.

### PERFORMANCE

- High efficiency, polycrystalline solar cells with high transmission and textured glass delivering a module efficiency of up to 10.9%, minimizing installation costs and maximizing the kWh output of your system per unit area.
- Power tolerance of +/-3% minimizing PV system mismatch losses.

### QUALITY AND RELIABILITY

- Industry leading in-house manufacturing of polysilicon, ingots, wafers, cells and modules ensures tight control of our material and production quality.
- Robust, corrosion resistant aluminum frame independently tested to withstand wind loads of 2.4 kPa and snow loads of 5.4 kPa ensuring a stable mechanical life for your modules.
- Module packaging optimized to protect product during transportation and minimize on-site waste.
- This type of module is commenly use for the small off-grid system.
- Manufacturing facility certified by TÜV Rheinland to ISO 9001:2008, ISO 14001:2004 and BS OHSAS 18001:2007.

### QUALIFICATIONS AND CERTIFICATES

IEC 61215, IEC 61730, UL 1703 and ULC 1703, UL Fire Safety Class C, ISO 9001:2008, ISO 14001:2004, BS OSHAS 18001:2007, SA 8000, PV Cycle

PV CYCLE

YINGLISOLAR.COM

# YGE 20 SERIES

## ELECTRICAL PERFORMANCE

| Electrical parameters at Standard Test Conditions (STC) | | | |
|---|---|---|---|
| Module name | | | YGE 20 |
| Module type | | | YL020P-17b 1/6 |
| Power output | $P_{max}$ | W | 20 |
| Power output tolerances | $\Delta P_{max}$ | % | +/- 3 |
| Module efficiency | $\eta_m$ | % | 10.9 |
| Voltage at $P_{max}$ | $V_{mpp}$ | V | 17.3 |
| Current at $P_{max}$ | $I_{mpp}$ | A | 1.16 |
| Open-circuit voltage | $V_{oc}$ | V | 21.3 |
| Short-circuit current | $I_{sc}$ | A | 1.32 |

STC: 1000W/m² irradiance, 25°C $T_{module}$, AM 1.5g spectrum according to EN 60904-3.
Ave. efficiency reduction of 5% at 200W/m² according to EN 60904-1.

| Electrical parameters at Nominal Operating Cell Temperature (NOCT) | | | |
|---|---|---|---|
| Power output | $P_{max}$ | W | 14.5 |
| Voltage at $P_{max}$ | $V_{mpp}$ | V | 15.8 |
| Current at $P_{max}$ | $I_{mpp}$ | A | 0.92 |
| Open-circuit voltage | $V_{oc}$ | V | 19.4 |
| Short-circuit current | $I_{sc}$ | A | 1.07 |

NOCT: open-circuit module operation temperature at 800W/m² irradiance, 20°C $T_{ambient}$, 1m/s wind speed.

## THERMAL CHARACTERISTICS

| | | | |
|---|---|---|---|
| Nominal operating cell temperature | NOCT | °C | 46 +/- 2 |
| Temperature coefficient of $P_{max}$ | γ | %/°C | -0.45 |
| Temperature coefficient of $V_{oc}$ | $\beta_{Voc}$ | %/°C | -0.37 |
| Temperature coefficient of $I_{sc}$ | $\alpha_{Isc}$ | %/°C | 0.06 |

## OPERATING CONDITIONS

| | |
|---|---|
| Max. system voltage | $50V_{DC}$ |
| Max. series fuse rating | 5A |
| Limiting reverse current | Do not apply external voltages larger than $V_{oc}$ of the module |
| Operating temperature range | -40 to 85°C |
| Max. static load, front (e.g., snow and wind) | 2400Pa |
| Max. static load, back (e.g., wind) | 2400Pa |
| Hailstone impact (hailstone diameter / impact velocity) | 25mm / 23m/s |

## CONSTRUCTION MATERIALS

| | |
|---|---|
| Front cover (material / thickness) | low-iron tempered glass / 3.2mm |
| Cell (quantity / type / dimensions) | 36 / multicrystalline / 78mm x 52mm |
| Encapsulant (material) | ethylene vinyl acetate (EVA) |
| Frame (material / color / anodization color) | anodized aluminum alloy / silver / clear |

- Due to continuous innovation, research and product improvement, the specifications in this product information sheet are subject to change without prior notice. The specifications may deviate slightly and are not guaranteed.
- The data do not refer to a single module and they are not part of the offer, they only serve for comparison to different module types.

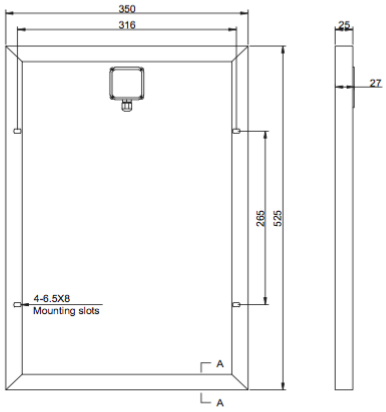
## GENERAL CHARACTERISTICS

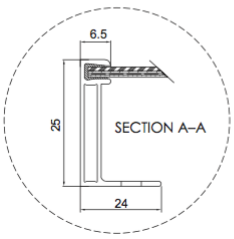
| | |
|---|---|
| Dimensions (length / width / height) | 525mm / 350mm / 25mm |
| Weight | 2.58 kg |

## PACKAGING SPECIFICATIONS

| | |
|---|---|
| Number of modules per box | 6 |
| Packaging box dimensions (length / width / height) | 570mm / 380mm / 210mm |

**Unit: mm**




Warning: Read the installation manual in its entirety before handling, installing, and operating Yingli Solar module.

Our Partners:

**Yingli Green Energy Holding Co. Ltd.**
service@yinglisolar.com
0086-312-8929802
YINGLISOLAR.COM

# APÊNDICE A - ESTRUTURA BASE PARA MÓDULOS

```
import time
from threading import Thread

MODULE_NAME = os.path.basename(__file__)
MODULE_NAME = MODULE_NAME[:MODULE_NAME.index(".")]

THREAD_REPEAT_TIME = 1
modulestart = False
modulerunning = False

def start(vars,*params):
    global modulestart
    if modulerunning == True:
        vars.msghandler(MODULE_NAME,"Error","Module already running")
    else:
        modulestart = True
        mainThread(vars).start()
        vars.msghandler(MODULE_NAME,"State","Module is running")
    return False    #isBusy = False

def stop(vars,*params):
    global modulestart
    if modulerunning == False:
        vars.msghandler(MODULE_NAME,"Error","Module already stopped")
    else:
        modulestart = False
        vars.msghandler(MODULE_NAME,"State","Module stopping")
        while modulerunning == True:
            time.sleep(1)
        vars.msghandler(MODULE_NAME,"State","Module stopped")
    return False    # isBusy=False

def check(vars,*params):
    if modulerunning == True:
    vars.msghandler(MODULE_NAME,"Check","Module is running")
    else:
        vars.msghandler(MODULE_NAME,"Check","Module is not running")

class mainThread(Thread):

    def __init__(self, vars):
        Thread.__init__(self)
        self.Vars = vars

        ##############################
        ##INSERT FUNCTIONALITIES HERE
        ##############################

    def run(self):
        global modulerunning

        ##############################
        ##INSERT FUNCTIONALITIES HERE
        ##############################

        while modulestart==True:
            modulerunning = True

            ##############################
            ##INSERT FUNCTIONALITIES HERE
            ##############################

            time.sleep(THREAD_REPEAT_TIME)
        modulerunning = False
```

# APÊNDICE B - CÓDIGO PIC

```
char uart_rcv[15];
int anglemoved, x;
char tmpchar[20], tmpchar2[20];
char uart_out[20];
char signal[2];
int OPERATIONMODE = 0;
int desiredangle[2], motor;
int running = 0;
int count=0;
int cyclelimit;

void movemotors(char string[]) {
   signal[0] = string[0];
   signal[1] = string[5];
   tmpchar[0]='\0';
   tmpchar2[0]='\0';
   for (motor=0;motor<=1;motor++) {
      desiredangle[motor]= (string[1+5*motor]-48)*100;
      desiredangle[motor] = desiredangle[motor] + (string[2+5*motor]-48)*10;
      desiredangle[motor] = desiredangle[motor] + (string[3+5*motor]-48);
      cyclelimit = desiredangle[motor];
      if(desiredangle[motor]!=0) {
         if(motor==0) {
            PORTC.F2 = 0;
         }
         else {
            PORTC.F2 = 1;
         }
         Delay_ms(100);
         count=0;
         if(signal[motor]=='-') {
            if(motor==0) {
               count=0;
               PORTB.F7=0;
               PORTB.F2=1;
               running = 1;
               while(running==1) {
                  Delay_ms(20);
               }
            }
            else {
               count=0;
               PORTB.F4=0;
               PORTB.F5=1;
               running = 1;
               while(running==1) {
                  Delay_ms(20);
               }
            }
            tmpchar[0]='P';
            tmpchar[1]='O';
            tmpchar[2]='S';
            if(motor==0){
               tmpchar[3]='1';
            }
            else {
               tmpchar[3]='2';
            }
            tmpchar[4]=':';
            tmpchar[5]='-';
            tmpchar[6]='\0';
            anglemoved = cyclelimit;
            IntToStrWithZeros(anglemoved, tmpchar2);
            strcat(tmpchar,tmpchar2);
         }
         else {
            if(motor==0) {
               count=0;
               PORTB.F7=1;
               PORTB.F2=0;
               running = 1;
               while(running==1) {
                  Delay_ms(20);
               }
            }
            else {
               count=0;
               PORTB.F4=1;
               PORTB.F5=0;
               running = 1;
               while(running==1) {
                  Delay_ms(20);
               }
```

```
                }
                tmpchar[0]='P';
                tmpchar[1]='O';
                tmpchar[2]='S';
                if(motor==0){
                    tmpchar[3]='1';
                }
                else {
                    tmpchar[3]='2';
                }
                tmpchar[4]=':';
                tmpchar[5]='+';
                tmpchar[6]='\0';
                anglemoved = cyclelimit;
                IntToStrWithZeros(anglemoved, tmpchar2);
                strcat(tmpchar,tmpchar2);
            }
            UART1_Write_Text(tmpchar);
        }
        else {
            if (motor==0) {
                strcpy(tmpchar,"POS1:+000000");
            }
            else {
                strcpy(tmpchar,"POS2:+000000");
            }
            UART1_Write_Text(tmpchar);
        }
        }
}

void sendadc() {
    int cnt, convert, tmpint;
    char output[20];
    char tmpchar[20];
    for(cnt=1;cnt<=6;cnt++) {
        tmpchar[0] = '\0';
        output[0] = '\0';
        convert = ADC_Read(cnt+1);
        output[0]='A';
        output[1]='D';
        output[2]='C';
        if(cnt==1) {
            output[3]='1';
        }
        if(cnt==2) {
            output[3]='2';
        }
        if(cnt==3) {
            output[3]='3';
        }
        if(cnt==4) {
            output[3]='4';
        }
        if(cnt==5) {
            output[3]='5';
        }
        if(cnt==6) {
            output[3]='6';
        }
        output[4]=':';
        output[5]=':';
        output[6]='0';
        output[7]='0';
        output[8]='0';
        tmpint = convert/100;
        output[9]= tmpint + '0';
        convert = convert - tmpint*100;
        tmpint = convert/10;
        output[10] = tmpint + '0';
        convert = convert - tmpint*10;
        output[11] = convert + '0';
        output[12]='\0';
        UART1_Write_Text(output);
    }
}

void main() {
    //ADC
    OPTION_REG.INTEDG = 1; // Set Rising Edge Trigger for INT
    INTCON.GIE = 1; // Enable The Global Interrupt
    INTCON.INTE = 1; // Enable INT
    ADCON1 = 0;
    ADC_Init();
    // UART
    UART1_Init(9600);
```

```
    // Reles
    TRISB.F7=0;   //Rele 1 A
    TRISB.F2=0;   //Rele 2 A
    TRISB.F4=0;   //Rele 1 B
    TRISB.F5=0;   //Rele 2 B
    PORTB.F7=1;
    PORTB.F2=1;
    PORTB.F4=1;
    PORTB.F5=1;
    // Rele para Interrupt
    TRISC.F2 = 0;
    PORTC.F2 = 0;
    // Sensores de curso
    TRISD.F1=1;
    TRISD.F2=1;
    UART1_Write_Text("Init");
    UART1_Write(13);
    UART1_Write(10);
    while(1) {
        if (UART1_Data_Ready()) {              // If data is received,
            UART1_Read_Text(uart_rcv, "$", 12);   // Read text until "$" appears
            if(strlen(uart_rcv)==9){
                movemotors(uart_rcv);
                sendadc();
            }
            UART1_Write_Text("END:00000000");
        }
    }
}

void interrupt() {
    INTCON.INTF=0; // Clear the interrupt 0 flag
    count = count + 1;
    if(count>cyclelimit) {
        running=0;
        PORTB.F7=1;
        PORTB.F2=1;
        PORTB.F4=1;
        PORTB.F5=1;
        count=0;
    }
}
```

## APÊNDICE C - CÓDIGO MÓDULO main.py

```
#!/usr/bin/env python
import time
import os

MODULE_NAME = os.path.basename(__file__)
### IMPORTA MODULOS
############################################################
shutdown = False
modules = []   #ARMAZENA OS MODULOS
modnames = []  #ARMAZENA OS NOMES DOS MODULOS
for file in os.listdir("./"):
   if file.endswith(".py"):
      file = file[:-3]
      if not file=='main':
         modnames.append(file)
         modules.append(__import__(file))
fullmodnames = ''
for x in range (0,len(modnames)):
   fullmodnames = fullmodnames + modnames[x] + ' ';
print "Modules added: %s" % fullmodnames

############################################################
##VARIAVEIS DO SISTEMA
############################################################
class Vars():
   power = 0.0
   ls1 = 0.0
   ls2 = 0.0
   ls3 = 0.0
   ls4 = 0.0
   pos1 = None
   pos2 = None
   desiredpos1 = None
   desiredpos2 = None
   opmode = 0 ## 0-MANUAL 1-AUTO SENSOR 2-AUTO MODEL
   outputmodules = []

   @staticmethod
   def addoutputmodule(module_name):
      for x in range (0,len(modules)):
         if module_name == modnames[x]:
            Vars.outputmodules.append(modules[x])
            Vars.msghandler(MODULE_NAME,"State","Output Module %s added sucessfully" % module_name)

   @staticmethod
   def removeoutputmodule(module_name):
      for x in range (0,len(modules)):
         if module_name == modnames[x]:
            Vars.outputmodules.remove(modules[x])
            Vars.msghandler(MODULE_NAME,"State","Output Module %s removed sucessfully" % module_name)


   @staticmethod
   def msghandler(module_name,msg_type,msg):
      str = "[%s][%s][%s]:%s" % (time.strftime("%Y-%m-%d %H:%M:%S", time.gmtime()),module_name,msg_type,msg)
      print "%s" % str
      if msg_type == 'Info' or msg_type =='State':
         str = "%s:%s" % (module_name,msg)
         for x in range (0,len(Vars.outputmodules)):
            function=getattr(Vars.outputmodules[x],"output")
            function(Vars,str)

   @staticmethod
   def cmdhandler(command,MODULE_NAME):
      global shutdown
      isBusy = False
      if command=="Shutdown":
         shutdown = stopall()

      else:
         try:
            ########################################################
            ### Cortar o comando no formato Classe_Funcao_Param1_Param2_ParamN
```

```
            ##############################################################
            count = command.count("_")
            param = []
            if count < 1:
               Vars.msghandler(MODULE_NAME,"Error","Command not found")
               print" Cmd not found" ##
            else:
               strclass = command[:command.index("_")]
               restofstring = command[command.index("_")+1:]
               if count == 1:
                  strdef = restofstring;
               else:
                  strdef = restofstring[:restofstring.index("_")]
                  while restofstring.count("_") > 0:
                     restofstring = restofstring[restofstring.index("_")+1:]
                     if restofstring.count("_") > 0:
                        param.append(restofstring[:restofstring.index("_")])
                     else:
                        param.append(restofstring)
            ###################################################################

            ###################################################################
            ### Executa o comando a partir das strings obtidas
            ###################################################################
            cmdfound = False
            for x in range (0,len(modules)):
               if strclass.lower() == modnames[x].lower():
                  cmdfound = True
                  isBusy = True
                  function=getattr(modules[x],strdef)
                  isBusy = function(Vars,param)
                  break
            if cmdfound is not True:
               Vars.msghandler(MODULE_NAME,"Error","Command not found")
         except:
            Vars.msghandler(MODULE_NAME,"Error","Command ERROR")
            isBusy=False
      while isBusy == True:
         time.sleep(1)
            ###############################################################

##############################################################
##INICIA TODOS OS MODULOS EM SEQUENCIA
##############################################################
for x in range (0,len(modnames)):
   Vars.msghandler(MODULE_NAME,"State","Starting module %s" % modnames[x])
   Vars.cmdhandler("%s_start" % modnames[x],MODULE_NAME)
##############################################################
def stopall ():
   try:
      for x in range (0,len(modnames)):
         Vars.cmdhandler("%s_stop" % modnames[x],MODULE_NAME)
      return True
   except:
      return False

##EXECUCAO DE COMANDOS
##############################################################
while True:
   if shutdown == True:
      break
   command = raw_input(">>")
   Vars.cmdhandler(command,MODULE_NAME)
##############################################################
```

## APÊNDICE D - CÓDIGO MÓDULO dbcom.py

```
import time
from threading import Thread
import MySQLdb
import os

MODULE_NAME = os.path.basename(__file__)
MODULE_NAME = MODULE_NAME[:MODULE_NAME.index(".")]
modulestart = False
modulerunning = False
THREAD_REPEAT_TIME = 5

def start(vars,*args):
    global modulestart
    if modulerunning == True:
        vars.msghandler(MODULE_NAME,"Error","Module already running")
    else:
        modulestart = True
        mainThread(vars).start()
        vars.msghandler(MODULE_NAME,"State","Module running")
    return False    #isBusy = False

def stop(vars,*args):
    global modulestart
    if modulerunning == False:
        vars.msghandler(MODULE_NAME,"Error","Module already stopped")
    else:
        modulestart = False
        vars.msghandler(MODULE_NAME,"State","Module is stopping")
        while modulerunning == True:
            time.sleep(1)
        vars.msghandler(MODULE_NAME,"State","Module is stopped")
    return False    # isBusy=False

def check(vars,*args):
    if modulerunning == True:
        vars.msghandler(MODULE_NAME,"Check","Module is running")
    else:
        vars.msghandler(MODULE_NAME,"Check","Module is not running")
    return False

class mainThread(Thread):

    def __init__(self, vars):
        Thread.__init__(self)
        self.Vars = vars
        self.db = connect()
    def run(self):
        global modulerunning
        while modulestart==True:
            modulerunning = True
            if self.Vars.pos1 is not None and self.Vars.pos2 is not None:
                insertdata(self.db,self.Vars.power,self.Vars.pos1, self.Vars.pos2, self.Vars.opmode)
            time.sleep(THREAD_REPEAT_TIME)
        modulerunning = False
        disconnect(self.db)

def connect():
    db = MySQLdb.connect("localhost","root","raspberry","tccdb" )
    return db

def insertdata( db, voltage, pos1, pos2, opmode):
    cursor = db.cursor()
    try:
        sql = "INSERT INTO voltdata (voltage, pos1, pos2, opmode,date,time) VALUES (%s,%s,%s,%s,curdate(),curtime())" %
(voltage,pos1,pos2,opmode)
        cursor.execute(sql)
        db.commit()
    except:
        db.rollback()

def disconnect (db):
    return db.close()
```

## APÊNDICE E - CÓDIGO MÓDULO uartcom.py

```
from threading import Thread
import random
import time
import os
import serial

MODULE_NAME = os.path.basename(__file__)
MODULE_NAME = MODULE_NAME[:MODULE_NAME.index(".")]
THREAD_REPEAT_TIME = 1
modulestart = False
modulerunning = False
readyfornextmessage = True
TIMEOUT = 60
COEF_MOTOR1 = 1.87
COEF_MOTOR2 = 2.04

def start(vars,*args):

    global modulestart
    if modulerunning == True:
        vars.msghandler(MODULE_NAME,"Error","Module already running")
    else:
        modulestart = True
        mainThread(vars).start()
        vars.msghandler(MODULE_NAME,"State","Module is running")
    return False    #isBusy = False

def stop(vars,*args):
    global modulestart
    if modulerunning == False:
        vars.msghandler(MODULE_NAME,"Error","Module already stopped")
    else:
        modulestart = False
        vars.msghandler(MODULE_NAME,"State","Module stopping")
        while modulerunning == True:
            time.sleep(1)
        vars.msghandler(MODULE_NAME,"State","Module stopped")
    return False    # isBusy=False

def check(vars,*args):
    if modulerunning == True:
        vars.msghandler(MODULE_NAME,"Check","Module is running")
    else:
        vars.msghandler(MODULE_NAME,"Check","Module in not running")
    return False

class mainThread(Thread):
    threadrunning = False
    def __init__(self, vars):
        Thread.__init__(self)
        self.Vars = vars
        self.connection = serial.Serial("/dev/ttyAMA0", baudrate=9600, timeout=3.0)
        self.incomingdata(self.connection,self.Vars).start()
    def run(self):
        global modulerunning
        global readyfornextmessage
        modulerunning = True
        timeoutcount = 0
        while modulestart==True:
            if readyfornextmessage is True and self.Vars.pos1 is not None and self.Vars.pos2 is not None and
self.Vars.desiredpos1 is not None and self.Vars.desiredpos2 is not None:
                timeoutcount = 0
                tmpchar = "%.3i" % (int)((self.Vars.desiredpos1-self.Vars.pos1)*COEF_MOTOR1)
                if len(tmpchar) == 3:
                    tmpchar = "+" + tmpchar
                tmpchar2 = "%.3i" % (int)((self.Vars.desiredpos2-self.Vars.pos2)*COEF_MOTOR2)
                if len(tmpchar2) == 3:
                    tmpchar2 = "+" + tmpchar2
                self.connection.write("%s,%s$" % (tmpchar,tmpchar2))
                self.Vars.msghandler(MODULE_NAME,"Data","%s,%s$" % (tmpchar,tmpchar2))
                readyfornextmessage = False
            time.sleep(THREAD_REPEAT_TIME)
            timeoutcount=timeoutcount+1
```

```
            if timeoutcount == TIMEOUT:
                readyfornextmessage = True
        while self.threadrunning==True:
            time.sleep(1)
        modulerunning = False

    class incomingdata (Thread):
        tmpvoltage = None
        tmpcurrent = None
        tmpls1 = None
        tmpls2 = None
        tmpls3 = None
        tmpls4 = None
        tmppos1 = None
        tmppos2 = None

        def __init__(self,connection,vars):
            Thread.__init__(self)
            self.connection=connection
            self.Vars = vars
        def run(self):
            global readyfornextmessage
            self.threadrunning = True

            while modulestart==True:
                rcv = ''
                    ### Recebendo valores
                while rcv == '' and modulestart is True:
                    rcv = self.connection.read(12)
                    time.sleep(0.5)
                self.Vars.msghandler(MODULE_NAME,"Data","Rcv: %s" % rcv)
                if rcv.__len__() is not 12:
                    self.Vars.msghandler(MODULE_NAME,"Error","Invalid message size")
                else:
                    if rcv[:4]=="ADC1":
                        self.tmpls1 = float(rcv[6:])*5/1024
                        print "Received LS1"
                    if rcv[:4]=="ADC2":
                        self.tmpls2 = float(rcv[6:])*5/1024
                        print "Received LS2"
                    if rcv[:4]=="ADC3":
                        self.tmpls3 = float(rcv[6:])*5/1024
                        print "Received LS3"
                    if rcv[:4]=="ADC4":
                        self.tmpls4 = float(rcv[6:])*5/1024
                        print "Received LS4"
                    if rcv[:4]=="ADC5":
                        self.tmpvoltage = float(rcv[6:])*5/1024
                        print "Received Voltage"
                    if rcv[:4]=="ADC6":
                        self.tmpcurrent = float(rcv[6:])*5/1024
                        print "Received Current"
                    if rcv[:4]=="POS1":
                        self.tmppos1 = self.Vars.pos1 + float(rcv[5:])/COEF_MOTOR1
                        print "Received POS1"
                    if rcv[:4]=="POS2":
                        self.tmppos2 = self.Vars.pos2 + float(rcv[5:])/COEF_MOTOR2
                        print "Received POS2"
                    if rcv=="<POS1ATZERO>":
                        self.Vars.pos1 = 0
                        print "Received POS1=0"
                    if rcv=="<POS2ATZERO>":
                        self.Vars.pos2 = 0
                        print "Received POS2=0"
                    if rcv[:3]=="END":
                        print "Received END"
                        if self.tmpvoltage is not None and self.tmpls1 is not None and self.tmpls2 is not None and self.tmpls3 is not None
and self.tmpls4 is not None and self.tmppos1 is not None and self.tmppos2 is not None:
                            self.Vars.power = self.tmpvoltage*4*self.tmpcurrent
                            self.Vars.ls1 = self.tmpls1
                            self.Vars.ls2 = self.tmpls2
                            self.Vars.ls3 = self.tmpls3
                            self.Vars.ls4 = self.tmpls4
                            self.Vars.pos1 = self.tmppos1
                            self.Vars.pos2 = self.tmppos2
                        else:
```

```
                self.Vars.voltage = None
                self.Vars.ls1 = None
                self.Vars.ls2 = None
                self.Vars.ls3 = None
                self.Vars.ls4 = None
                self.Vars.pos1 = None
                self.Vars.pos2 = None
            self.tmpvoltage = None
            self.tmpls1 = None
            self.tmpls2 = None
            self.tmpls3 = None
            self.tmpls4 = None
            self.tmppos1 = None
            self.tmppos2 = None
            readyfornextmessage = True
        ####
self.threadrunning = False
```

## APÊNDICE F - CÓDIGO MÓDULO tcpcom.py

```
import time
import socket
import sys
from threading import Thread
import os

MODULE_NAME = os.path.basename(__file__)
MODULE_NAME = MODULE_NAME[:MODULE_NAME.index(".")]
modulestart = False
modulerunning = False
SERVER_IP = "192.168.2.10"
SERVER_PORT = 12001
connection = None
waitingconnection = False

def start(vars,*args):
    global modulestart
    if modulerunning == True:
        vars.msghandler(MODULE_NAME,"Error","Module already running")
    else:
        vars.addoutputmodule(MODULE_NAME)
        modulestart = True
        mainThread(vars).start()
        vars.msghandler(MODULE_NAME,"State","Module is running")
    return False    #isBusy = False

def stop(vars,*args):
    global modulestart
    if modulerunning == False:
        vars.msghandler(MODULE_NAME,"Error","Module is already stopped")
    else:
        vars.removeoutputmodule(MODULE_NAME)
        modulestart = False
        if waitingconnection is True:
            socket.socket(socket.AF_INET, socket.SOCK_STREAM).connect( (SERVER_IP, SERVER_PORT))
        vars.msghandler(MODULE_NAME,"State","Module is stopping")
        while modulerunning == True:
            time.sleep(1)
        vars.msghandler(MODULE_NAME,"State","Module is stopped")
    return False    # isBusy=False

def check(vars,*args):
    if modulerunning == True and waitingconnection == False:
        vars.msghandler(MODULE_NAME,"Check","Module is running and connected")
    elif waitingconnection == True:
        vars.msghandler(MODULE_NAME,"State","Module is waiting for connection")
    else:
        vars.msghandler(MODULE_NAME,"State","Module is not running")
    return False

class mainThread(Thread):
    def __init__(self, vars):
        Thread.__init__(self)
        self.Vars = vars

    def run(self):
        global modulerunning
        global connection
        global waitingconnection
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server_address = (SERVER_IP, SERVER_PORT)
        self.Vars.msghandler(MODULE_NAME,"Connection","Starting up on %s port %s" % (SERVER_IP,SERVER_PORT))
        sock.bind(server_address)
        sock.listen(True)
        while modulestart==True:
            self.Vars.msghandler(MODULE_NAME,"Connection","Waiting for a connection")
            waitingconnection = True
            modulerunning = True
            connection, client_address = sock.accept()
            waitingconnection = False
            if modulestart is False:
                connection.close()
            else:
```

```
            try:
                self.Vars.msghandler(MODULE_NAME,"Connection","Connection from %s %s" % client_address)
                # Receive the data in small chunks and retransmit it
                while True:
                    data = connection.recv(35)
                    if not data == "":
                        self.Vars.msghandler(MODULE_NAME,"Data",".%s." % data)
                        if data.count("$") > 0:
                            data = data[:data.index("$")]
                            self.Vars.cmdhandler(data,MODULE_NAME)
                        else:
                            self.Vars.msghandler(MODULE_NAME,"Info","Message doesnt have the terminator $")
                    else:
                        connection.close()
                        connection = None
                        break
            except:
                connection.close()
            finally:
                self.Vars.msghandler(MODULE_NAME,"Connection","Connection closed")

        modulerunning = False

def output(vars,str):
    global connection
    if not connection == None:
        connection.sendall("%s\n" % str)

def setzero(vars, *args):
    vars.pos1 = 0
    vars.pos2 = 0
    vars.desiredpos1 = 0
    vars.desiredpos2 = 0

def timetracking(vars, *args):
    vars.opmode = 2

def sensortracking(vars, *args):
    vars.opmode = 1

def manualtracking(vars, *args):
    vars.opmode = 0

def setsensors(vars, *args):
    for arg in args:
        if len(arg)==4:
            vars.ls1 = float(arg[0])
            vars.ls2 = float(arg[1])
            vars.ls3 = float(arg[2])
            vars.ls4 = float(arg[3])

def setpos(vars, *args):
    for arg in args:
        if len(arg)==2:
            vars.pos1 = float(arg[0])
            vars.pos2 = float(arg[1])

def setdesiredpos(vars, *args):
    for arg in args:
        if len(arg)==2:
            print "2"
            vars.desiredpos1 = float(arg[0])
            vars.desiredpos2 = float(arg[1])
       if len(arg)==3:
            print"3"
            vars.desiredpos1 = float(arg[0])
            vars.desiredpos2 = float(arg[1])
            vars.opmode = int(arg[2])

def checkvariables(vars, *args):
    if vars.power is not None:
        vars.msghandler(MODULE_NAME,"Info","Power: %f" % vars.power)
    else:
        vars.msghandler(MODULE_NAME,"Info","Power is null")

    if vars.ls1 is not None:
```

```
    vars.msghandler(MODULE_NAME,"Info","Ls1: %f" % vars.ls1)
else:
    vars.msghandler(MODULE_NAME,"Info","Ls1 is null")

if vars.ls2 is not None:
    vars.msghandler(MODULE_NAME,"Info","Ls2: %f" % vars.ls2)
else:
    vars.msghandler(MODULE_NAME,"Info","Ls2 is null")

if vars.ls3 is not None:
    vars.msghandler(MODULE_NAME,"Info","Ls3: %f" % vars.ls3)
else:
    vars.msghandler(MODULE_NAME,"Info","Ls3 is null")

if vars.ls4 is not None:
    vars.msghandler(MODULE_NAME,"Info","Ls4: %f" % vars.ls4)
else:
    vars.msghandler(MODULE_NAME,"Info","Ls4 is null")

if vars.pos1 is not None:
    vars.msghandler(MODULE_NAME,"Info","Pos1: %f" % vars.pos1)
else:
    vars.msghandler(MODULE_NAME,"Info","Pos1 is null")

if vars.pos2 is not None:
    vars.msghandler(MODULE_NAME,"Info","Pos2: %f" % vars.pos2)
else:
    vars.msghandler(MODULE_NAME,"Info","Pos2 is null")

if vars.desiredpos1 is not None:
    vars.msghandler(MODULE_NAME,"Info","Desiredpos1: %f" % vars.desiredpos1)
else:
    vars.msghandler(MODULE_NAME,"Info","Desiredpos1 is null")

if vars.desiredpos2 is not None:
    vars.msghandler(MODULE_NAME,"Info","Desiredpos2: %f" % vars.desiredpos2)
else:
    vars.msghandler(MODULE_NAME,"Info","Desiredpos2 is null")

if vars.opmode is not None:
    vars.msghandler(MODULE_NAME,"Info","Opmode: %i" % vars.opmode)
else:
    vars.msghandler(MODULE_NAME,"Info","Opmode is null")
```

## APÊNDICE G - CODIGO MÓDULO suntracker.py

```
import time
from threading import Thread
import datetime
import math
import os

MODULE_NAME = os.path.basename(__file__)
MODULE_NAME = MODULE_NAME[:MODULE_NAME.index(".")]
THREAD_REPEAT_TIME = 1
modulestart = False
modulerunning = False


def start(vars,*args):
    global modulestart
    if modulerunning == True:
        vars.msghandler(MODULE_NAME,"Error","Module already running")
    else:
        modulestart = True
        mainThread(vars).start()
        while modulerunning == False:
            time.sleep(1)
        vars.msghandler(MODULE_NAME,"State","Module is running")
    return False    #isBusy = False

def stop(vars,*args):
    global modulestart
    if modulerunning == False:
        vars.msghandler(MODULE_NAME,"Error","Module already stopped")
    else:
        modulestart = False
        while modulerunning == True:        ## Espera ate que o modulo seja desligado
            time.sleep(1)
        vars.msghandler(MODULE_NAME,"State","Module is stopped")
    return False    # isBusy=False

def check(vars,*args):
    if modulerunning == True and waitingconnection == False:
        vars.msghandler(MODULE_NAME,"Check","Module is running and connected")
    else:
        vars.msghandler(MODULE_NAME,"State","Module is not running")
    return False

class mainThread(Thread):

    def __init__(self, vars):
        Thread.__init__(self)
        self.Vars = vars
    def run(self):
        At=0
        A=0
        Ht=0
        pi=3.14159
        t=0
        N=0
        N0=0
        Ed=0
        Et=0
        Cosfi=0
        Sd=0
        S=0
        F=0
        Jd=0
        Jf=0
        Sr=0
        tal=0
        Fr=0
        Hr=0
        fi=0
        SinHr=0
        CosA=0
        deltaGMT=-3
        LSTM=0
```

```
        EoT=0
        B=0
        TC=0
        LST=0
        Lat=-25
        Long=-49
        LT=0
        gama=0
        HRA=0
        alfa=0
        Sunrise=0
        Sunset=0
        Azi=0
        teta=44.6
        D=0
        Aux_Den=0
        Aux_Num=0
        global modulerunning
        modulerunning = True
        while modulestart==True:
            if self.Vars.opmode == 1 and self.Vars.pos1 is not None and self.Vars.pos2 is not None:
                ##logic for following from sensor values
                #print "\nopmode:%i,ls1:%.2f,ls2:%.2f,ls3:%.2f,ls4:%.2f,pos1:%f,pos2:%f,dpos1:%f,dpos2:%f" %
(opmode,ls1,ls2,ls3,ls4,pos1,pos2,dpos1,dpos2)

                avt = (self.Vars.ls1 + self.Vars.ls2) / 2   # average value top
                avd = (self.Vars.ls3 + self.Vars.ls4) / 2   # average value down
                avl = (self.Vars.ls1 + self.Vars.ls3) / 2   # average value left
                avr = (self.Vars.ls2 + self.Vars.ls4) / 2   # average value right

                dvert = avt - avd  # check the difference of up and down
                dhoriz = avl - avr  # check the difference of left and right

                if dvert < 0.2 or dvert > -0.2:
                    nextpos1 = self.Vars.pos1 + dvert*3
                else:
                    if dvert < 0.5 or dvert > -0.5:
                        nextpos1 = self.Vars.pos1 + dvert*7
                    else:
                        nextpos1 = self.Vars.pos1 + dvert*12

                if dvert < 0.15 or dvert > -0.15:
                    nextpos2 = self.Vars.pos2 + dhoriz*3
                else:
                    if dvert < 0.4 or dvert > -0.4:
                        nextpos2 = self.Vars.pos2 + dhoriz*7
                    else:
                        nextpos2 = self.Vars.pos2 + dhoriz*12

                if self.Vars.opmode == 1:
                    self.Vars.desiredpos1 = nextpos1
                    self.Vars.desiredpos2 = nextpos2


            if self.Vars.opmode == 2 and self.Vars.pos1 is not None and self.Vars.pos2 is not None:
            ##logic for following from localtime

                now = datetime.datetime.now()
                hour = now.hour + 0.0
                min = now.minute + 0.0
                sec = now.second + 0.0
                d = datetime.datetime.now().timetuple().tm_yday
                #Local standard time Meridian
                LSTM = 15*deltaGMT

                # Equation of Time
                B=(0.986301369863)*(d-81)
                B=B*0.0174532925
                EoT=9.87*math.sin(2*B)-7.53*math.cos(B)-1.5*math.sin(B)
                #EoT=3.493;
                #Time Correction Factor
                TC=4*(Long-LSTM)+EoT

                #Time to Decimal Time
                LT=hour+min/60+sec/3600
```

```
            #Local Solar Time
            LST=LT+(TC/60)

            #hour angle
            HRA=15*(LST-12)
                                    #Declination angle
            gama=math.asin(math.sin(23.55*0.0174532925)*math.sin(B))
            gama=gama/0.0174532925
            D=23.45*math.sin(0.986301369863*(284+d)*0.0174532925)
                                    #Elevation angle

alfa=math.asin((math.sin(gama*0.0174532925)*math.sin(Lat*0.0174532925))+(math.cos(gama*0.0174532925)*math.cos(Lat*0.
0174532925)*math.cos(HRA*0.0174532925)))
            alfa=alfa/0.0174532925
            Sd=-math.tan(Lat*0.0174532925)*math.tan(gama*0.0174532925)
            Sunrise=12-((math.acos(Sd*0.0174532925))/(15*0.0174532925))-(TC/60)
            Sunset=12+((math.acos(Sd*0.0174532925))/(15*0.0174532925))-(TC/60)
                                    #Altitude angle

teta=math.asin((math.cos(Lat*0.0174532925)*math.cos(gama*0.0174532925)*math.cos(HRA*0.0174532925))+(math.sin(Lat*0.
0174532925)*math.sin(gama*0.0174532925)))
            teta=teta/0.0174532925
                                    #Azimute angle
            Aux_Den=(math.cos(teta*0.0174532925)*math.cos(Lat*0.0174532925))
            Aux_Num=((math.sin(teta*0.0174532925)*math.sin(Lat*0.0174532925))-math.sin(D*0.0174532925))
            Aux_conta=Aux_Num/Aux_Den

            if Aux_conta > 1:
               Aux_conta=1
            if Aux_conta < -1:
               Aux_conta=-1
            Azi=(math.acos(Aux_conta))/0.0174532925
            if HRA < 0:
            #Azi=180-Azi
            #Azi=90-Azi
               Azi=90-Azi
               Azi=abs(Azi)
            else:
               Azi=270-Azi
            if self.Vars.opmode ==2:
               self.Vars.desiredpos1 = alfa
               self.Vars.desiredpos2 = Azi
            #print "\n Local Solar Time: %.3f \n Local Time : %.3f\n Equation of Time : %.3f\n Elevation(Altitude) angle:  %.3f\n
Declination angle:  %.3f\n Hour angle:  %.3f\n Azimuth angle:  %.3f\n  " % (LST, LT, EoT, alfa, D, HRA, Azi)
         ### End of localtime logic
         time.sleep(THREAD_REPEAT_TIME)
      modulerunning = False

############################################################
```

# APÊNDICE H - CÓDIGOS PHP

dbconnect.php:

```
<?php
$host = "localhost"; //servidor de banco de dados
$user = "root"; //usuario do banco de dados
$senha_connect = "raspberry"; //senha do banco de dados
$db = "tccdb"; //banco de dados

$sqlconnect=mysql_connect($host, $user, $senha_connect);
if(!$sqlconnect)
   die("Nao conectou com o banco de dados.");
$sqldb=mysql_select_db("$db",$sqlconnect);
if(!$sqldb)
   die("Nao foi possivel selecionar a base de dados.");
mysql_set_charset('utf8', $sqlconnect);
?>
```

index.php:

```
<?php
   date_default_timezone_set('America/Sao_Paulo');

   ## Conexao com o banco de dados
   include "dbconnect.php";
   ##

   ## Obtencao de valores via GET
   $_select = $_GET['select'];
   $_maxpos1 = $_GET['maxpos1'];
   $_maxpos2 = $_GET['maxpos2'];
   $_minpos1 = $_GET['minpos1'];
   $_minpos2 = $_GET['minpos2'];
   $_opmode = $_GET['opmode'];
   $_days = $_GET['days'];
   $_hours = $_GET['hours'];
   $_mode = $_GET['mode'];
   $_bdate = $_GET['bdate'];
   $_edate = $_GET['edate'];
   $_btime = $_GET['btime'];
   $_etime = $_GET['etime'];
   ##
   if ($_select=='') {
      $_select = 'power,pos1,pos2,ls1,ls2,ls3,ls4,opmode,date,time';
   }
   if ($_mode=='') {
      $_mode = 'json';
   }
   ## parsear $_select para selecionar os valores a serem retornados
   $_select = " " . $_select;
   $select = array();
   if (strpos($_select,'power') != false) {
      array_push($select,"power");
   }
   if (strpos($_select,'pos1') != false) {
      array_push($select,"pos1");
   }
   if (strpos($_select,'pos2') != false) {
      array_push($select,"pos2");
   }
   if (strpos($_select,'ls1') != false) {
      array_push($select,"ls1");
   }
   if (strpos($_select,'ls2') != false) {
      array_push($select,"ls2");
   }
   if (strpos($_select,'ls3') != false) {
      array_push($select,"ls3");
   }
   if (strpos($_select,'ls4') != false) {
      array_push($select,"ls4");
   }
   if (strpos($_select,'opmode') != false) {
      array_push($select,"opmode");
   }
   if (strpos($_select,'date') != false) {
```

```
    array_push($select,"date");
}
if (strpos($_select,'time') != false) {
    array_push($select,"time");
}
###

## Criacao da string de selecao dos dados
$sql = "SELECT * FROM suntrackerdata";
$count = 0;
if($_minpos1 != null) {
    if($count==0){
        $sql = $sql . " WHERE ";
    }
    else {
        $sql = $sql . " AND ";
    }
    $sql = $sql."pos1>'" . $_minpos1 . "'";
    $count = $count + 1;
}
if($_maxpos1 != null) {
    if($count==0){
        $sql = $sql . " WHERE ";
    }
    else {
        $sql = $sql . " AND ";
    }
    $sql = $sql."pos1<'" . $_maxpos1 . "'";
    $count = $count + 1;
}
if($_minpos2 != null) {
    if($count==0){
        $sql = $sql . " WHERE ";
    }
    else {
        $sql = $sql . " AND ";
    }
    $sql = $sql."pos2>'" . $_minpos2 . "'";
    $count = $count + 1;
}
if($_maxpos2 != null) {
    if($count==0){
        $sql = $sql . " WHERE ";
    }
    else {
        $sql = $sql . " AND ";
    }
    $sql = $sql."pos2<'" . $_maxpos2 . "'";
    $count = $count + 1;
}
if($_opmode != null) {
    if($count==0){
        $sql = $sql . " WHERE ";
    }
    else {
        $sql = $sql . " AND ";
    }
    $sql = $sql."opmode='" . $_opmode . "'";
    $count = $count + 1;
}
if($_bdate != null) {
    if($count==0){
        $sql = $sql . " WHERE ";
    }
    else {
        $sql = $sql . " AND ";
    }
    $sql = $sql."date>='" . $_bdate . "'";
    $count = $count + 1;
}
if($_edate != null) {
    if($count==0){
        $sql = $sql . " WHERE ";
    }
    else {
        $sql = $sql . " AND ";
```

```
    }
    $sql = $sql."date<='" . $_edate . "'";
    $count = $count + 1;
}
if($_btime != null) {
    if($count==0){
        $sql = $sql . " WHERE ";
    }
    else {
        $sql = $sql . " AND ";
    }
    $sql = $sql."time>='" . $_btime . "'";
    $count = $count + 1;
}
if($_etime != null) {
    if($count==0){
        $sql = $sql . " WHERE ";
    }
    else {
        $sql = $sql . " AND ";
    }
    $sql = $sql."time<='" . $_etime . "'";
    $count = $count + 1;
}
if($_days != null) {
    if($count==0){
        $sql = $sql . " WHERE ";
    }
    else {
        $sql = $sql . " AND ";
    }
    $date = date_create(date('Y-m-d H:i:s'));
    $minus_date = $_days . " days";
    date_sub($date, date_interval_create_from_date_string($minus_date));
    $date = date_format($date, 'Y-m-d');
    $sql = $sql."date>='" . $date . "'";
    $count = $count + 1;
}
if($_hours != null) {
    if($count==0){
        $sql = $sql . " WHERE ";
    }
    else {
        $sql = $sql . " AND ";
    }
    $date = date_create(date('Y-m-d H:i:s'));
    $minus_date = $_hours . " hours";
    date_sub($date, date_interval_create_from_date_string($minus_date));
    $date = date_format($date, 'H:i:s');
    $sql = $sql."time>='" . $date . "'";
    $count = $count + 1;
}
$sql = $sql . " order by date desc, time desc";
###

## Modos de output
if($_mode=='json') {
    header("Content-Type: text/html;  charset=ISO-8859-1",true);
    $count=0;
    $query = mysql_query($sql);
    while($row = mysql_fetch_array($query,MYSQL_ASSOC)){
        if ($count==0){
            echo '{"data":[';

        }
        if ($count!=0) {
            echo ',';
        }

        for($x = 0; $x < count($select); $x++) {
            if($x==0)
                echo '{';
            else
                echo ',';

            if ($select[$x]=='date' || $select[$x]=='time')
```

```
                echo ''.$select[$x].'':''.$row[$select[$x]].'';
            else
                echo ''.$select[$x].'':'.$row[$select[$x]];


            ## Final do dataset
            if($x==count($select)-1)
                echo '}';
            ###
        }
        $count = 1;
    }
    if ($count!=0) {
        echo ']}';
    }
}
if($_mode=='xml') {
    header("Content-Type: application/xml");
    $count=0;
    $query = mysql_query($sql);
    while($row = mysql_fetch_array($query,MYSQL_ASSOC)){
        if ($count==0){
            echo '<?xml version="1.0" encoding="utf-8"?>';
            echo '<sunfollowerdata>';
        }
        echo '<data>';
        for($x = 0; $x < count($select); $x++) {
            echo '<'.$select[$x].'>'.$row[$select[$x]].'</'.$select[$x].'>';
        }

        echo '</data>';
        $count = 1;
    }
    if ($count!=0) {
        echo '</sunfollowerdata>';
    }
}
if($_mode=='excel') {
    $filename = "sunfollower_" . date('Ymd') . ".csv";
    header("Content-Disposition: attachment; filename=\"$filename\"");
    header("Content-Type: application/vnd.ms-excel");
    $query = mysql_query($sql);
    $count=0;
    while($row = mysql_fetch_array($query,MYSQL_ASSOC)){
        if($count==0) {
            for($x = 0; $x < count($select); $x++) {
                if($x!=0)
                    echo ';';
                echo $select[$x];
            }
            $count=1;
        }
        echo "\r\n";
        for($x = 0; $x < count($select); $x++) {
            if($x!=0)
                    echo ';';
                echo $row[$select[$x]];
            }
        }
    }
    ###
?>
```