WSN Lab 2019, Group 1: Solar Pro

Karthik Sukumar
Electrical and Computer Engineering
Technical University of Munich
Munich, Germany
karthik.sukumar@tum.de

Johannes Machleid
Electrical and Computer Engineering
Technical University of Munich
Munich, Germany
johannes.machleid@tum.de

***Abstract*—The source of future energy production has to be and is undoubtedly renewable and environmentally friendly. Solar power is one of the most readily available energy sources in almost all parts of the world. Although solar power is ubiquitous, certain physical and technological limitations allow for a maximum efficiency factor of 37% (and thats for commercially avaiable high end solar cells). This implies that only 37% of the sun's energy that is captured by the solar cell can be converted to useful electrical energy. This paper focuses on using Wireless Sensor Networks (WSN) to utilise the maximum possible energy of the solar cells without any further losses by aligning the solar panel orthogonal to the sunrays depending on the daytime.**

I. Introduction

The energy generated by the solar panels is highly dependent on the angle of incidence of the sunrays on the panel.

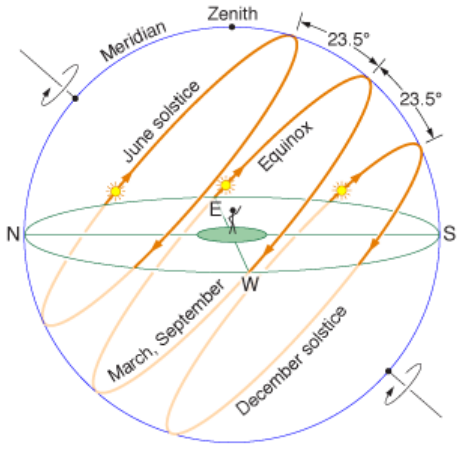


Fig. 1: Suns path in Winter and Summer. Source: [**?**]

The sun's path in the horizon is dynamic and dependent on the time of the year. In the northern hemisphere, the sun is higher in the sky during summer and lower in the winter. In the summer the sun rises in the north east and sets in the north west. Whereas in winter, the sun rises in the south east and sets in the south west as it can be seen in Fig. **??**

As we can see from Fig. **??**, consequent of the sun's path in the sky, the angle of incidence of the sun's rays on the panel changes not only during the day but as well as during the months of the year.

In order to track the sun's rays, so that the solar panel is always perpendicular to it, we need a dynamic system that

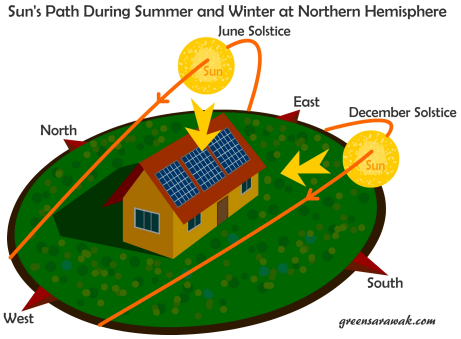


Fig. 2: Angle of incidence of the suns rays on the panel Source: [**?**]

adjusts to the E-W changes in the suns angle during the different times of the day as well the N-S tracking during the changes in the months.

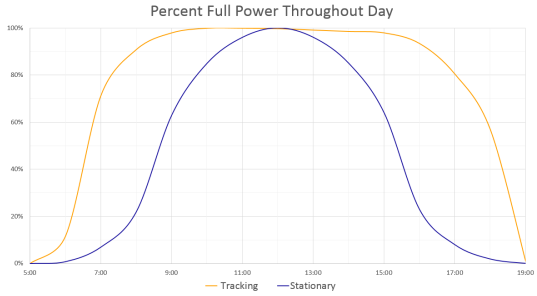


Fig. 3: Comparison of energy produced by a tracking vs non tracking system Source: [**?**]

II. Application Set up

Fig. **??** is a comparison of the percentage of total power generated between a tracking and a non-tracking panel. It can be seen that there are definitely gains to be made from employing a tracking panel. Our proof of concept therefore showcases the potential to deploy a wireless sensor network to not only control the tracking of solar panels but log data at the same time. In order to achieve this we employ three different types of sensors and eight wireless motes, of which one is set up as the base station connected to a desktop computer and the seven others deployed around the basestation to function as sensor and panel control motes.

Since only four digital servo motors are available, not every mote in the field is able to align the solar panel according to the sun's rays. Never the less we can still measure and transmit sensor values. A possible setup with a multihop communication towards the base station is depicted in Fig. **??**.

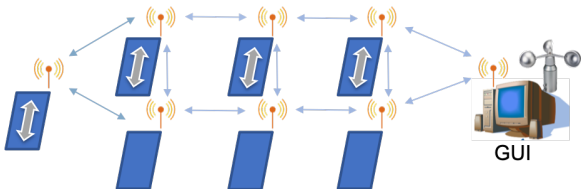


Fig. 4: Possible application setup with seven sensor motes and a base station ensuring multihop communication.

A. Sensors

Described below are the different types of sensors used in our application along and their uses:

- Digital Servo Motor - HS-422 [**?**]
  A servo motor is a cheap rotary actuator which is connected to the mote via three wires: Ground (GND), Voltage (VDD) and Signal. The servo angle can be set via the signal wire by using a PWM-Signal. Most servos expect to see a pulse every 20ms [**?**]. The correspondence between pulsewidth and servo angle is depicted in formula (**??**). The servos will be used to change the angle of solar panels.

$$pw = 1.0ms + 1.0ms \cdot \left(\frac{\alpha}{\alpha_{max}}\right) \tag{1}$$

- Light Sensors [**?**]
  The light sensors are used in this proof of concept to measure the luminosity and thus simulate the power output of the solar panel, since no real solar panels are available. The actual luminosity in lux is not calculated, since only the qualitative ADC 12-bit values of the sensors are of interest.

- Wind Speed Sensor [**?**]
  The wind speed sensor, also called anemometer, is directly connected to the base station. It measures the wind speed at all times. In the case of excessive wind speeds, the panels will have to be stowed at a safe angle.
  The anemometer works with a simple reed contact and a magnet attached to the rotating axis [**?**]. Every time the magnet passes the reed contact, a digital HIGH is sent to the base station, which is processed by the base station using interrupts. The measured interval $t_{measure}$ together with the interrupt count n_{ticks} delivers the actual wind speed *ws* in km/h after formula (**??**).

$$ws = \frac{n_{ticks}}{t_{measure}} \cdot 2.4\frac{km/h}{tick} \tag{2}$$

- Zolertia RE-mote platform [**?**]
  The Zolertia RE-mote is a wireless module designed to be small, consume very little power, stay affordable and be easy to deploy in large quantities. In general, these type of devices are known as *motes*. These motes are the brain of the wireless sensor network and are running Contiki as operating system.

B. Motes

We are using eight motes in total out of which

- 7x Zolertia REmotes are used as nodes attached to solar panels
- 1x Zolertia REmote is used as a base station

1) Base station: The base station is connected to a desktop computer via a serial link and feeds the GUI with information. It runs a different code compared to the panel motes as it has to account for interfacing with the computer and collect data from the wind speed sensor at the same time.

2) Panel Motes: The panel motes run a more simplified subset of the base station code. They only react to unicast and broadcast messages and they resond based on the packet type.

III. Network description and Routing Technique

In this proof of concept we are demonstating an energy harvesting WSN which provides information on the status of the solar panel along with the ability to control the angle of the panel in 1 axis. In a real world scenario it is intended that the mote [**?**] will be connected to the solar panel and be able to draw minimal power required for the functioning of the mote.

A. Network

Considering our application and the reliability as one of the factors, we decided that a centralised base station with distributed routing table. A Centralised polling based quering scheme suits our application the most. This also eliminates the need for synchronisations (which are potentially required in a distributed system) and makes the implementation simpler.
The routes are first discovered during the network discovery phase and the routing tables are exchanged as described in **??**

B. Base Station State Machine

In our application we have 5 modes of operation, which are

- IDLE
- NETWORKDISCOVERY
- PATHMODE
- UNICASTMODE
- EMERGENCY

1) IDLE: When the device comes out of reset this is the state, the mote gets into. In this state there are no broadcast or unicast messages that are sent out by the base station.

2) NETWORKDISCOVERY: This state is entered upon a user button press or a trigger from the GUI. The network discovery is used to obtain the network graph with the base station as the sink for the information

3) PATHMODE: This state is a transition from where the base station polls the nodes for their total path to the base station

4) UNICASTMODE: In this state the nodes are polled in a round-robin manner and when the base station queries the information it sends the servo angle data along with the query packet

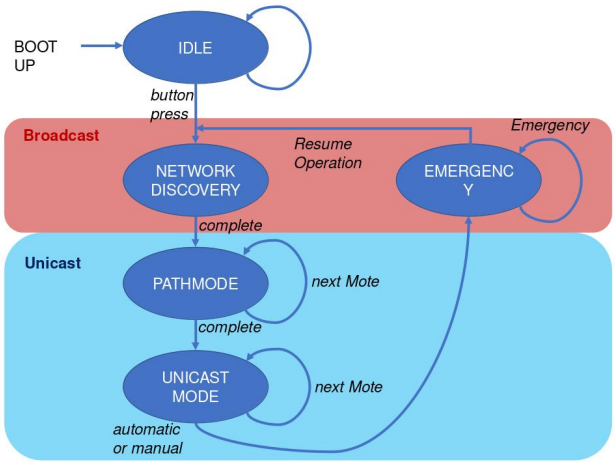


Fig. 5: Base Station State Machine

5) EMERGENCY:

C. Cost

An important consideration is the cost for a route. There are many factors that can determine the calculation of a route cost. It could include battery state, transmission power, hop count, RSSI, etc... , in our application we have directly equated hop count with the cost.

D. Network Discovery

This is the most critical phase of the state machine. The Network discovery here is initiated by a user button press which makes it easier to demonstrate our proof of concept. Furthermore it can also be triggered from the GUI as explained in **??**. Any mote can initiate a network discovery which then sends a broadcast packet. This node's complete routing table is sent as a payload in this broadcast message to exchange with others. This is received by all the neighbours and compared with their own tables. If there are updates to be made i.e. routes with lower cost from the senders routing table, the receiver updates their table and retransmits the broadcast after a short delay. This is illustrated in fig

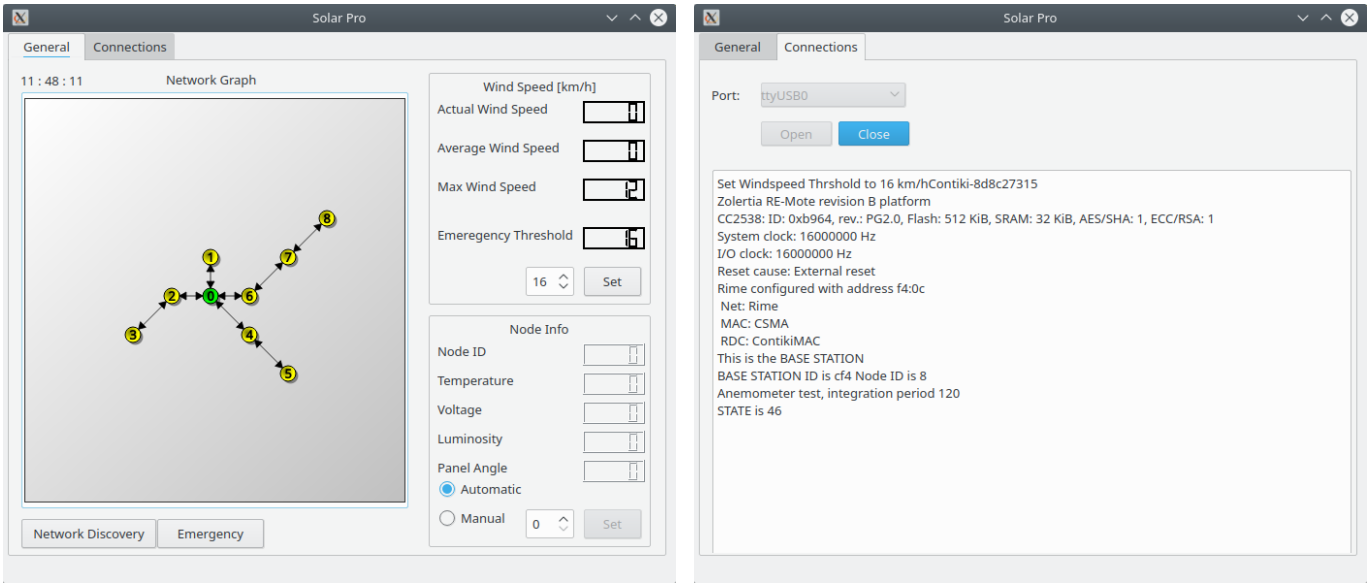
E. Packet Types

We have 2 different communication schemes which can be found below

- Broadcast
  - EMERGENCY - This broadcast packet is sent out in the emergency phase which stows the panels at a predefined safe angle
  - NETDISC - This packet type is sent out during the network discovery phase which contains sender's routing tables
- Unicast
  - PATH - This unicast packet type is sent during the PATHMODE of operation as described in **??**
  - ACK - This unicast packet type is an ackledgement responce sent during the PATHMODE. This contains the hop history of from the base station to the target node
  - UNICAST - This unicast packet type is sent during the UNICASTMODE of operation as described in **??**. In this mode the nodes are polled in round-robin fashion at regular intervals and this goes on uninterrupted unless an Emergency is triggered

IV. Graphical User Interface (GUI)

The graphical user interface (GUI) is a desktop application programmed with the aid of the QT Creator and focuses on displaying information about the motes operating in the field and allows the user to interact with a desired mote. The computer running the GUI is directly connected to the base station mote via a serial link.
The GUI basically consists of the two tabs "General" and "Connections" as shown in Fig. **??** and Fig. **??**.
The "General" tab shows a network graph of the wireless sensor network and the system time. Communicating sensor motes are depicted with a yellow circle and a number inside, whereas the base station is shown centered as green circle. The doublesided arrows depict the wireless communication route and therefore the network topology. It does not show possible wireless communication between sensor motes, because only the shortest paths towards the base station as sink in the network is of importance. The two buttons "Network Discovery" and "Emergency" in the bottom left corner enable the user to trigger these two operational states manually.
Two info boxes aligned to the right side of the window show the sensor information of the motes. The upper info box shows the values of the anemometer connected to the base station, such as the actual wind speed, the average windspeed, the maximum windspeed and the emergency threshold, at which the solar panels are aligned to a predefined safety angle. The GUI also allows to choose a user defined emergency threshold, which is set by pressing the "Set" button next to the number field. The default value is 16 km/h.
The lower info box shows the sensor information of the selected mote. A mote is selected by clicking on a mote in the network graph. The particular information such as the Node ID, the temperature, voltage, luminosity and panel angle are presented. For maintenance reasons it is also possible to operate the selected mote by clicking on the "Manual" button. It is then possible to enter an angle between 0 and 180 degrees to manually align the selected solar panel by clicking the "Set" button next to the number field.
The second tab of the GUI allows the user to connect and disconnect the serial link to the base station via selecting the respective port and clicking the buttons "Open" to connect or "Close" to disconnect. The textbox aligned to the bottom of the tab shows possible debug information from the base station.



(a) GUI General Tab

(b) GUI Connections Tab

Fig. 6: Graphical User Interface of the Solar Pro Application

V. Application Specific Features

The MAC layer is not modified and runs the Contiki standard configuration ”ContikiMAC” according to [**?**].

VI. Future work and Optimisations

Our implementation does not focus on low latency so this could be a possible improvement. One way of improving latency can be achieved by using hash tables instead of array based routing tables. This reduces the complexity from O(n) to a constant time. Considering that for every UNICAST transmission routing table walk is necessary, this could save time and power. Further ideas to advance the application are a sun tracking algorithm which steers the solar panel according to sensor values. Another useful feature would be to integrate data visualization of the sensor values as a graph showing the history of e.g. the last 24 hours.

VII. Conclusion

The power efficiency of solar panels highly depends on the inclanation angle of the sunrays. By aligning the solar panel orthogonally to the sunrays depending on the daytime it is possible to increase its energy efficiency. In this application a wireless sensor network has been employd to control the solar panel angle via a digital servo and measure amount of light hitting the panel with a light sensor. Furthermore the temperature of the controlling mote and its battery voltage is read out.
The application setup consists of seven sensor motes, of which four motes operate a digital servo. One mote functions as base station and delivers the sensor information to a desktop application with a GUI.
The network is set up by first initialising a network discovery routine. All motes broadcast and collect information about their neighbors to set up their very own distance vector table. After completion, the base station reaches out to each mote with a unicast request to collect the shortest path through their respective acknowledge.
With this information the base station is able to poll the sensor informations of each node by cycling through the network.
The network graph and sensor information are visually presented in the graphical user interface, which also allows to operate the servo position manually, set a wind speed emergency threshold or simply trigger an emergency case manually. It is also possible to manually restart the network discovery.

References

[1] K. Sukumar, “Servo Motor HS-422,” https://www.multiplex-rc.de/produkte/112422-servo-hs-422, 2008, [Online; accessed 19-July-2008].

[2] Jameco.com, “How servo motors work,” accessed 2019-08-14. [Online]. Available: https://www.jameco.com/jameco/workshop/howitworks/how-servo-motors-work.html

[3] A. Egger, “Arduino anemometer bauen,” 2018, accessed 2019-08-14. [Online]. Available: https://www.aeq-web.com/anemometer-mit-dem-arduino-bauen/

[4] K. Sukumar, “Zolertia Remote,” https://github.com/Zolertia/Resources/wiki/RE-Mote, 2008, [Online; accessed 19-July-2008].

[5] A. Dunkels, “The contikimac radio duty cycling protocol,” 2011.