EJS with Phidgets Lab: Controls for Solar Demonstration Project

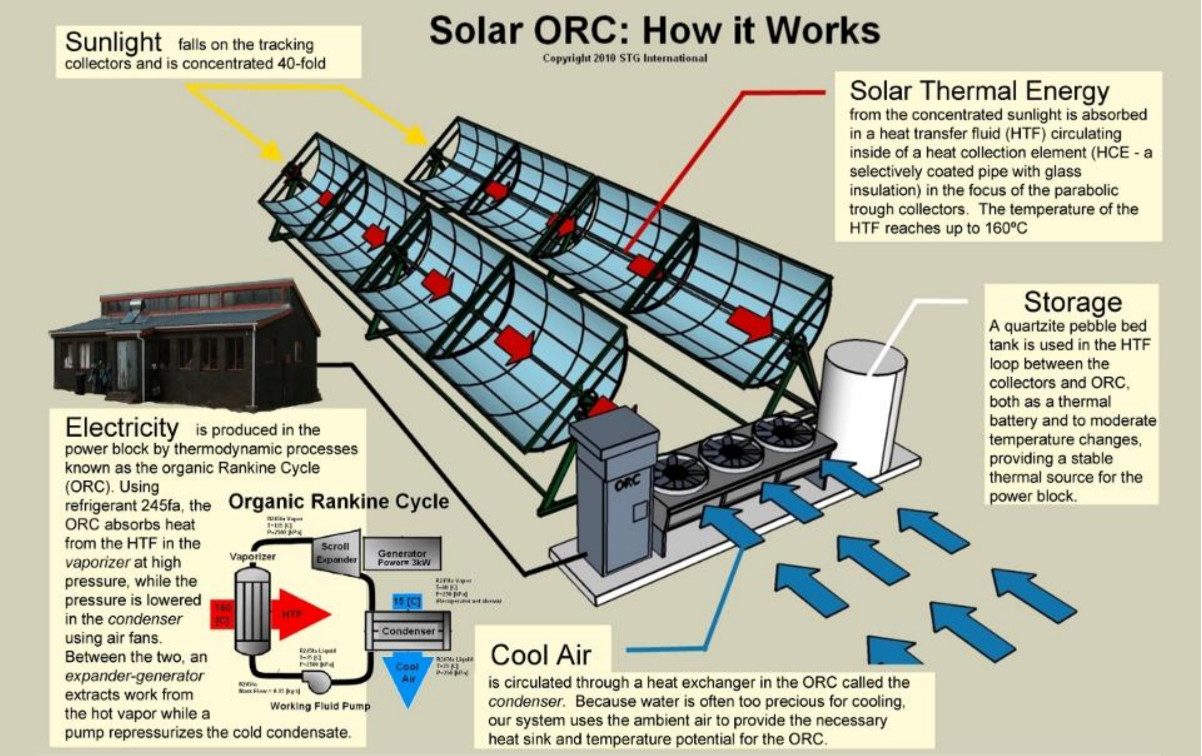

Solar ORC: How it Works
Copyright 2010 STG International
Sunlight falls on the tracking collectors and is concentrated 40-fold
Solar Thermal Energy from the concentrated sunlight is absorbed in a heat transfer fluid (HTF) circulating inside of a heat collection element (HCE - a selectively coated pipe with glass insulation) in the focus of the parabolic trough collectors. The temperature of the HTF reaches up to 160°C
Storage A quartzite pebble bed tank is used in the HTF loop between the collectors and ORC, both as a thermal battery and to moderate temperature changes, providing a stable thermal source for the power block.
ORC
Electricity is produced in the power block by thermodynamic processes known as the organic Rankine Cycle (ORC). Using refrigerant 245fa, the ORC absorbs heat from the HTF in the vaporizer at high pressure, while the pressure is lowered in the condenser using air fans. Between the two, an expander-generator extracts work from the hot vapor while a pump repressurizes the cold condensate.
Organic Rankine Cycle
Vaporizer
Scroll Expander
Generator Power= 3kW
160 [C]
HTF
15 [C]
Condenser
Cool Air
Working Fluid Pump
Cool Air is circulated through a heat exchanger in the ORC called the condenser. Because water is often too precious for cooling, our system uses the ambient air to provide the necessary heat sink and temperature potential for the ORC.

Figure 1: Schematic of the solar collector and Rankine Cycle Engine as designed by STG International to bring power to clinics in Lesotho, Africa. Image credits: http://www.stginternational.org/how-it-works.html

Overview:

There is a solar demonstration project just outside Galbraith Marine Science Lab, Eckerd College, St Petersburg, FL. It is part of a project to provide power to medical clinics in Lesotho, Africa by STG International using solar heating of essentially anti-freeze (http://www.stginternational.org). It includes a number of engineering design concerns. The purpose of this lab is to see how you can apply some physics and build a model system (a scaled down version) of the two of these elements using EJS (Easy Java Simulations).

1) Temperature sensor
2) Tracking mirror

1) Temperature Sensor:

To electronically sense temperature, we use thermocouples—wires of two different materials connected together. These wires have a voltage difference between them that depends on the surrounding temperature. You will measure the voltage from a thermocouple at different temperatures and construct a calibration curve to determine the temperature, then you will build a warning system if the temperature goes over a certain point.

2) Tracking Mirror:

The parabolic mirrors need to follow the sun so they are on a mount that can rotate. You will measure the period of a rotating object and set up a motor to track the rotation. The motors we will use are “servo motors” that are designed to move to a given angle given a specific input voltage (or voltage pulse).

If you haven’t used Easy Java Simulations before, please use the following tutorials and worksheets before moving to the individual tasks.

Easy Java Simulations:

You have been using simulations written using Easy Java Simulations (EJS). Now you are going to edit one that will talk to your thermocouple or motor (Phidget). Start with the Tutorial found at

http://www.compadre.org/OSP/tutorials/EJS_Tutorial/

and start at **4. Modifying a Simulation**. Watch through the “view” section to get a sense of what an EJS simulation looks like.

Next, read through the sheet on EJS instructions and answer the two Quick Checks:

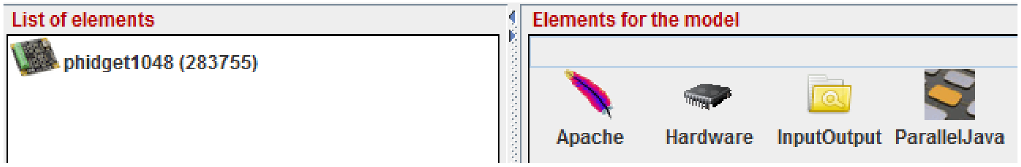
Quick Check 1:

Quick Check 2:

Temperature Sensor:

Task 1: Connect your Thermocouple to one of the Channels on the Sensor Board (white to G and red to the number to the left). What number did you choose? ______

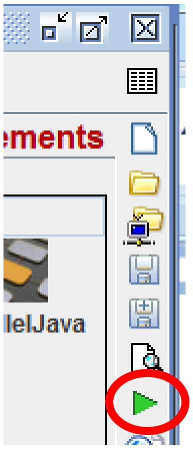
Start EJS and open the file: `PhidgetTemperatureExample`. Click to the `Model`and go to the `Elements` page.



Double-click on phidget1048 and input the serial number of your Servo board (it is probably NOT 2483755 as in the program). Your serial number _________________

Click to `Variables`:
What is the value of `channel`? _____
Change this (if needed) to match the number you connected your thermocouple to (listed above).

Run (push the green arrow) the program and describe what it does:



Do the temperature values seem reasonable (for both Celsius and Fahrenheit)?

What happens when you push the “TooHot” checkbox?

Task 2: From the Evolution page, modify the equation that displays the temperature in Fahrenheit (temperatureF) so that it displays the correct value.

Briefly explain what you did:

Task 3: Put in a warning control that flashes the “TooHot” warning when the temperature gets greater than a set value (you decide the value). Hint: You will need an `if statement`.

Describe what you did:

Servo Motor:

You will use the second hand of the clock in the room as the “sun” you need to match.

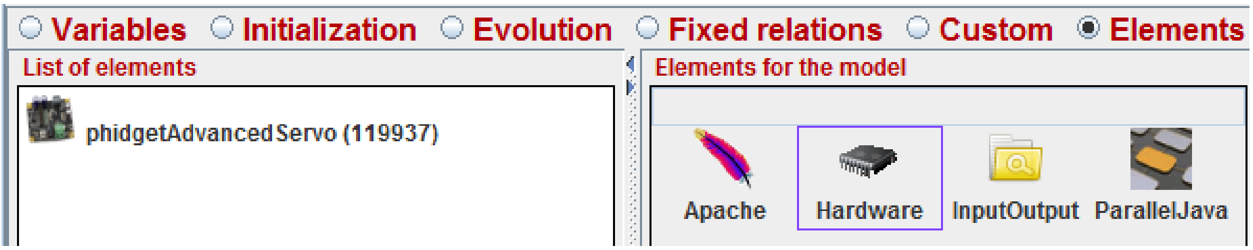
How many degrees/second does the motor need to move to “follow” the “sun”?___________

The motor can move through no more than 180o. Decide how far your “mirrors” (if connected to the motor) are going to move each “day” (each minute):

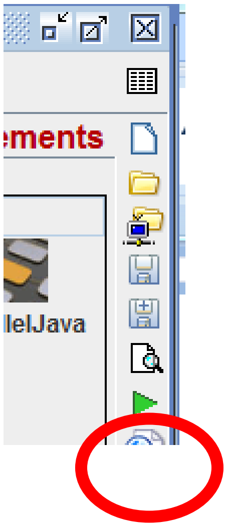
Degrees your motor will move: ______________

Time your motor will be off (or at the initial position): _____________

Task 1: Start EJS and open the file: `PhidgetMotorControl`. Click to the `Model`and go to the `Elements` page.



Double-click on AdvancedServo and input the serial number of your Servo board (it is probably NOT 119937 as in the program). Your serial number _________________



Run (push the green arrow) the program and describe what it does:

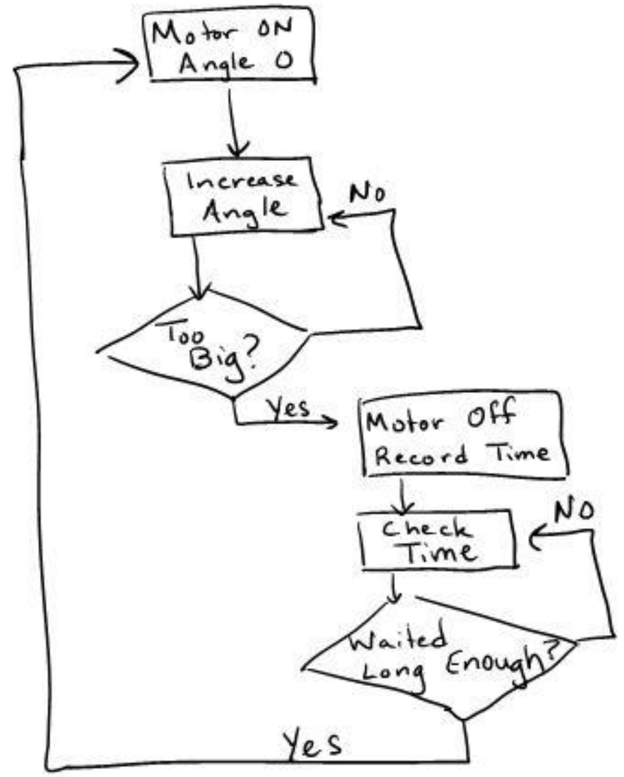
What does the plot show?

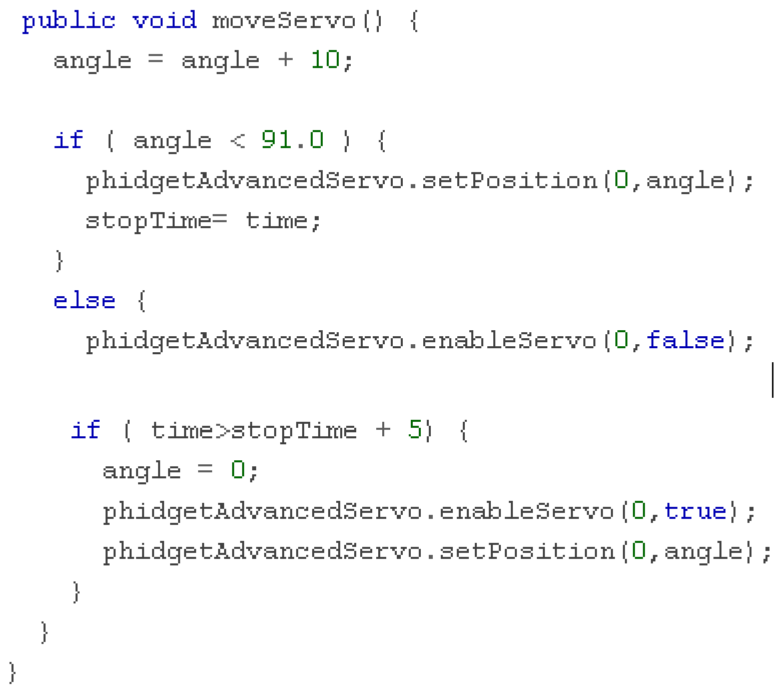
Explore the following parts of the `Model`:

<u>**`Variables:`**</u> What `Variables` does the `Model` use (list them)?

<u>**`Evolution:`**</u> What does this page tell the program to do (the `Evolution` page is executed every second)?

Custom: This is the main program that gets used so you need to look at in detail to figure out what to change. Programs are sometimes described by flow chart diagrams that show decision points. Below you have a flowchart and the code that executes it. Identify (match) the parts of the code that do each of the things in this flow chart[1]:



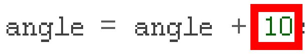


```
public void moveServo() {
  angle = angle + 10;

  if ( angle < 91.0 ) {
    phidgetAdvancedServo.setPosition(0,angle);
    stopTime= time;
  }
  else {
    phidgetAdvancedServo.enableServo(0,false);

     if ( time>stopTime + 5) {
       angle = 0;
       phidgetAdvancedServo.enableServo(0,true);
       phidgetAdvancedServo.setPosition(0,angle);
     }
   }
}
```

Task 2: What do you need to change (in the boxes below) for your motor motion to match the second hand of the clock (our "sun")?

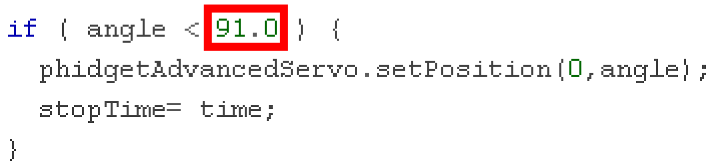
1) 

```
angle = angle + 10
```

What did you change it to and why?

[1] To get a sense of flowcharts (in the form of a flowchart) see: http://xkcd.com/518/.

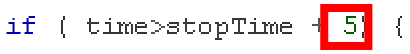
2)



```
if ( angle < 91.0 ) {
  phidgetAdvancedServo.setPosition(0,angle);
  stopTime= time;
}
```

What did you change it to and why?

3)



```
if ( time>stopTime + 5) {
```

What did you change it to and why?

Task 3: If the system gets too hot (as measured by the temperature sensor), add to your program something that will take your assembly to position 0 if the system overheats. Show your code below (Hint: you can put an `if` statement on the Evolution page):