SASTA OLIPHANT SCIENCE AWARDS

SOUTH AUSTRALIAN SCIENCE TEACHERS ASSOCIATION

Prize Winner

# Computer Programming, Apps & Robotics

## Year 7-8


Joshua Cartledge

Glenunga International High School


Government of South Australia
Department for Education

SASTA South Australian Science Teachers Association

ROWE SCIENTIFIC PTY. LTD. www.rowe.com.au

# Description of Entry:

The Rainwater Tank Monitor (RTM) is a monitoring device that utilises the eLabtronics STEMSEL microchip kit and an ultrasonic sensor to reliably monitor the level of a rainwater tank. This data is presented on a webpage that can be viewed on any device. The RTM also has a temperature sensor and fan in the internals to regulate its temperature on hot days, as well as an in-built text-to-speech (TTS) alert system to warn the user when their tank is low.

## Purpose of Entry:

The RTM's main purpose would be for farmers. Some farmers in the Australian Outback have rainwater tank that are hundreds of kilometres away that must be checked often for leaks or damage. This wastes hour of the farmer's time driving for long periods just to do a 5-minute check-up on their tanks. This device could be installed on those tanks and would enable the farmer to check the water levels at a glance. This would save the owner time and money.

## Scientific Principles Demonstrated:

The Rainwater Tank Monitor utilizes a device called the "HC-SR04 Ultrasonic Sensor" that operates using the science of ultrasonics.


4 pins are on the device. The VCC and GND pins are + and – respectively. The TRIG pin is an input and the ECHO pin is an output.

When a signal (short 10 µS long burst of power) is received by the sensor on the TRIG pin, the sensor will then emit 8 rapid pulses of ultrasonic noise. This noise is at a frequency of 40KHz, and so is outside the human hearing range. Otherwise, it would get annoying really quickly! Anyway, as soon as it emits those pulses, the ECHO pin will output power as a signal. It continues to do this until the sound reflects off the object and returns to the sensor or until 38 ms has passed (if the signal did not reflect). The sensor is then able to receive this signal as the distinct 8-pulse pattern can be distinguished from the background ultrasonic noise. It then turns off the pin.

What a microchip can do is measure the length of the signal from the ECHO pin, then using that length calculate the distance from the sensor to the object. This is done through simple math. Distance = speed * time, and we know that our time was the length of the signal (we'll use 700 µS as an example) and that the

speed is the speed of sound. We will convert the speed of sound to cm/µS (which is around 0.034 cm/µS) so that we can insert that into the equation.

D = 0.034 * 700

D = 23.8 cm

But wait, the time measured by the ultrasonic sensor was the time that it took for the ultrasonic pulse to travel both to the object and back to the sensor, so we must divide the result by two.

D = (0.034 * 700) / 2

D = 11.9 cm

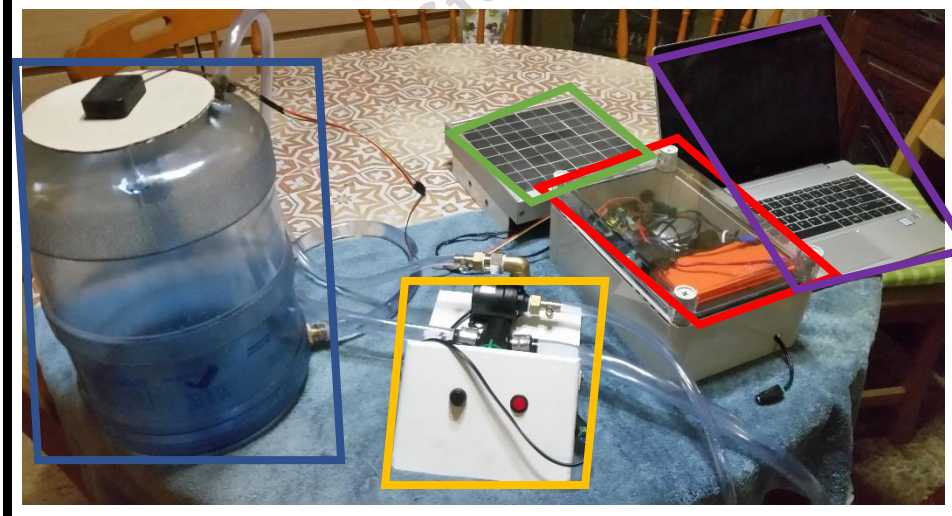
Therefore, the distance from the sensor to the object is 11.9cm.

In the case of the RTM, the sensor is mounted at the top of the tank and will output the distance from the top of the tank to the water level. This may be useless on its own, but when the user inputs the height of the tank into the webpage it is but a simple subtraction problem to find exactly how full the tank is. With the diameter of the tank, the volume of the tank and the volume of water can also be calculated. This is the basis behind the RTM.



## Hardware (with photos):

The RTM demonstration model is made of three components: the tank, the control panel, and the main box.



Components:

Blue: Rainwater Tank

Yellow: Control Panel for tank

Red: Main Box

Green: Solar Panel

Purple: Laptop

The tank consists of a water cooler bottle with pipes coming in and out to control the water level. The lowest pipe is connected to a valve to lower the level of water present in the tank, while the upper pipe is connected to a pump to raise the water level. There is an ultrasonic sensor mounted to the top of the tank.

Parts:

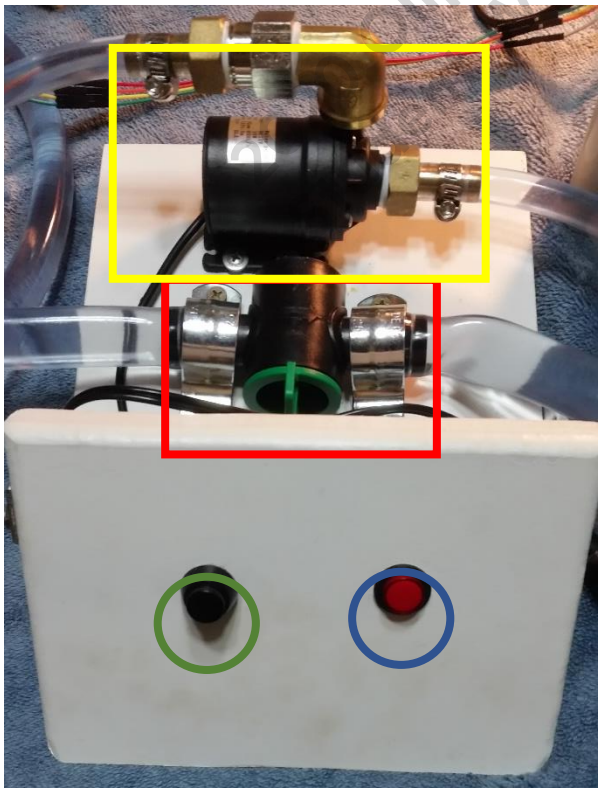
Red: Ultrasonic Sensor

Green: Inflow Pipe

Yellow: Outtake Pipe



The control panel has the valve and pump mounted on its back, with two buttons on its face. The right turns on and off the pump, while the left was meant to control the electronic valve, which unfortunately had to be replaced with a



Parts:

Blue: Pump controller button

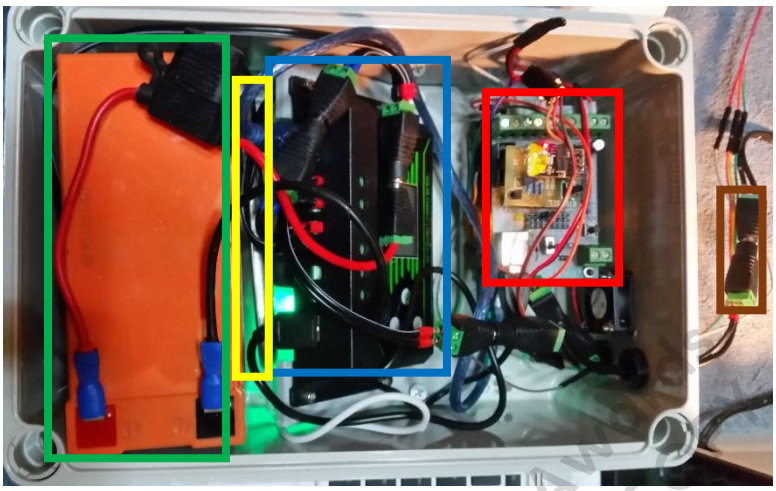
Green: Valve open/close (inoperable)

Red: Manual valve

Yellow: pump and attached hose connectors

manual one. This part of the model is merely for demonstration purposes (to control the tank level) and so is not a part of the RTM.

The final and most complex component of the RTM is the main box. The box contains several sub-components, shown in the diagram below:



Red: The aforementioned eLabtronics STEMSEL microchips. There are two mounted on top of each other in this project. They get the input from the ultrasonic sensor and temperature sensor and sends those values to the runlinc Wi-Fi chip mounted on the top board. This chip will connect to the phone to load the webpage. It also receives any signals to turn on/off the fan.

Yellow: A old phone that acts as a mini website hosting device for the chip. It connects to the chip and loads the webpage, then hosts that website via a website hosting service (Pagekite) through a data connection.

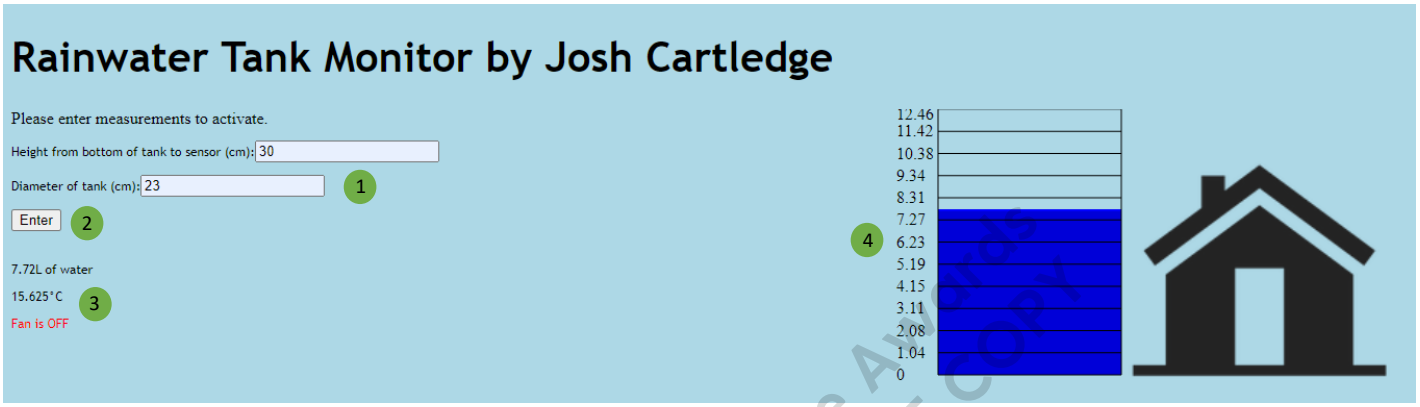
Blue: The Solar Charge Controller (SCC). It takes power from the solar panel (brown plug) and charges the battery. It also charges the phone and powers the microchips from the battery. This unit and the microchips are mounted on a custom 3D printed baseplate I designed that is screwed down inside the box.

Green: The 12V 9Ah Battery that powers the system.

# The Program:

Short Disclaimer: The following information refers to V1.3 of the RTM code and as such is subject to change between time of report submission and Open Day.

The Webpage:




This is what is displayed on the webpage.

1. Input boxes for the measurements of the tank. These measurements are used to calculate the water level and exact volume of water in the tank.
2. The "Enter" button confirms these measurements.
3. Statistics on amount of water present in tank, temperature of the internals and current fan state.
4. The graphic of the tank with accompanying measurements and house. The blue water level rises and falls with the tank's level.



The program operates through 3 simple steps.

1. Collect input from sensor and add it to a list.
2. Average the current list (no matter the length) and display the result on the graphic.
3. Check if any alert thresholds have been met and if so, trigger the alerts.

The list caps at length 500, and once it does so, the next sensor reading will override the first list element, and then the second, and so on. This means that the output is both smooth and relatively accurate.

# The Code:

Since runlinc is a web-based system, it utilises the JS (Javascript), HTML (Hyper-Text Markup Language) and CSS (Cascading Style Sheets) webkit of languages. The “JS Loop” section is part of the runlinc interface (online demo link here) and it essentially just repeatedly runs that section of code.

CSS:

```
h1 {
   font-family: "Trebuchet MS", Helvetica, sans-serif;
   font-size: 40px;
}

p {
   font-family: "Trebuchet MS", Helvetica, sans-serif;
   font-size: 12px;
}

html {
   background-color: lightblue;
}
```



The CSS simply changes the font, size, and colour of the different text types as well as sets the background to light blue.

HTML:

```
<h1>Rainwater Tank Monitor by Josh Cartledge</h1>
<svg width="500" height="300" style="float:right">
<image href="https://i.ibb.co/BrFtbWf/house.png" x="216" y="7"
height="295" width="265"/>
```

Displays the title, initialises the canvas and displays the picture of the house

```
<rect id="waterlevel2" x="40" y="0" height="0" width="176"
fill="rgb(0,0,216)"/>
<line id="waterlevel" x1="40" y1="0" x2="216" y2="0"
style="stroke:rgb(0,0,255);stroke-width:2" />
<line id="line1" x1="216" y1="255" x2="40" y2="255"
style="stroke:rgb(0,0,0);stroke-width:1" />
<text id="zero" x="0" y="260">0</text>
<line id="line2" x1="216" y1="233.75" x2="40" y2="233.75"
style="stroke:rgb(0,0,0);stroke-width:1" />
<text id="twenty" x="0" y="238.75">20</text>
<line id="line3" x1="216" y1="212.5" x2="40" y2="212.5"
style="stroke:rgb(0,0,0);stroke-width:1" />
<text id="fourty" x="0" y="217.5">40</text>
<line id="line4" x1="216" y1="191.25" x2="40" y2="191.25"
style="stroke:rgb(0,0,0);stroke-width:1" />
<text id="sixty" x="0" y="196.25">60</text>
<line id="line5" x1="216" y1="170" x2="40" y2="170"
style="stroke:rgb(0,0,0);stroke-width:1" />
<text id="eighty" x="0" y="175">80</text>
<line id="line6" x1="216" y1="148.75" x2="40" y2="148.75"
style="stroke:rgb(0,0,0);stroke-width:1" />
<text id="hundred" x="0" y="153.75">100</text>
<line id="line7" x1="216" y1="127.5" x2="40" y2="127.5"
style="stroke:rgb(0,0,0);stroke-width:1" />
<text id="onetwenty" x="0" y="132.5">120</text>
<line id="line8" x1="216" y1="106.25" x2="40" y2="106.25"
style="stroke:rgb(0,0,0);stroke-width:1" />
<text id="oneforty" x="0" y="111.25">140</text>
<line id="line9" x1="216" y1="85" x2="40" y2="85"
style="stroke:rgb(0,0,0);stroke-width:1" />
<text id="onesixty" x="0" y="90">160</text>
<line id="line10" x1="216" y1="63.75" x2="40" y2="63.75"
style="stroke:rgb(0,0,0);stroke-width:1" />
<text id="oneeighty" x="0" y="68.75">180</text>
<line id="line11" x1="216" y1="42.5" x2="40" y2="42.5"
style="stroke:rgb(0,0,0);stroke-width:1" />
```

Draws all the lines on the tank, as well as the text labels (to be changed by the JS).

```
<text id="twohundred" x="0" y="47.5">216</text>
<line id="line12" x1="216" y1="21.25" x2="40" y2="21.25"
style="stroke:rgb(0,0,0);stroke-width:1" />
<text id="twotwenty" x="0" y="26.25">220</text>
<line id="line13" x1="216" y1="0" x2="40" y2="0"
style="stroke:rgb(0,0,0);stroke-width:1" />
<text id="totalvolume" x="0" y="10">240</text>
<line id="line14" x1="216" y1="0" x2="216" y2="255"
style="stroke:rgb(0,0,0);stroke-width:1" />
<line id="line14" x1="40" y1="0" x2="40" y2="255"
style="stroke:rgb(0,0,0);stroke-width:1" />
</svg>
<p>Please enter measurements to activate.</p>
<br>
<p>Height from bottom of tank to sensor (cm):<input
id="height"></input></p>
<p>Diameter of tank (cm):<input id="diameter"></input></p>
<button onclick="Enter()">Enter</button>
<br><br>
<p id="value">0</p>
<p id="temperature">0°C</p>
<p id="fanStatus">Fan is OFF</p>
<p id="alerts" style="color:red;"></p>
```

Displays the inputs and “Enter” button.

Displays the statistics.



JS (lines are soft-wrapped at end of page):

```
var l1 = document.getElementById("waterlevel");
var r1 = document.getElementById("waterlevel2");   (these two can now be
referenced in the JS later to set the water level)
var d = 0;   (diameter)
var h = 0;   (height)
var i = 0;   (used as a counter)
var StoredVals = []   (the averaging list)
var average = 0;
var sensor = 0;
var output = 0;
var Subtractor = 0;
var Setter = 0;
```

Declares Variables (Additional info may be next to some, in parentheses)

```
var TotalVol = 1;
var WaterVol = 1;
var Activate = 0;
var temp = 0;   (temperature)
var speakAgain = 1;  (a flag used to ensure the TTS does not repeat incessantly)
var msg1 = new SpeechSynthesisUtterance('Water is running low!');
var msg2 = new SpeechSynthesisUtterance('Tank is almost full!');
msg1.rate = 0.7; // 0.1 to 10
msg2.rate = 0.7; // 0.1 to 10
const arrAvg = arr => arr.reduce((a,b) => a + b, 0) / arr.length
function Enter() {
  Activate = 1;
  var input1 = document.getElementById("diameter");
  var input2 = document.getElementById("height");
  d = input1.value;
  h = input2.value;
  TotalVol = (Math.round(((Math.pow(d / 2, 2)) *
3.14 * h) / 10) / 100).toString();
  document.getElementById('totalvolume').innerHTML = TotalVol;
  var Subtractor = TotalVol / 12;
  var Setter = TotalVol;
  Setter = Setter - Subtractor;
  document.getElementById('twotwenty').innerHTML = (Math.round(Setter *
100) / 100).toString();
  Setter = Setter - Subtractor;
  document.getElementById('twohundred').innerHTML = (Math.round(Setter *
100) / 100).toString();
  Setter = Setter - Subtractor;
  document.getElementById('oneeighty').innerHTML = (Math.round(Setter *
100) / 100).toString();
  Setter = Setter - Subtractor;
  document.getElementById('onesixty').innerHTML = (Math.round(Setter * 100)
/ 100).toString();
  Setter = Setter - Subtractor;
  document.getElementById('oneforty').innerHTML = (Math.round(Setter *
100) / 100).toString();
  Setter = Setter - Subtractor;
```

Setup TTS alerts.

Function to get the average of a list/array.

Function when the button is pressed. Sets d and h variables from user input.

```
 document.getElementById('onetwenty').innerHTML = (Math.round(Setter *
100) / 100).toString();
 Setter = Setter - Subtractor;
 document.getElementById('hundred').innerHTML = (Math.round(Setter *
100) / 100).toString();
 Setter = Setter - Subtractor;
 document.getElementById('eighty').innerHTML = (Math.round(Setter * 100) /
100).toString();
 Setter = Setter - Subtractor;
 document.getElementById('sixty').innerHTML = (Math.round(Setter * 100) /
100).toString();
 Setter = Setter - Subtractor;
 document.getElementById('fourty').innerHTML = (Math.round(Setter * 100) /
100).toString();
 Setter = Setter - Subtractor;
 document.getElementById('twenty').innerHTML = (Math.round(Setter * 100)
/ 100).toString();
 Setter = Setter - Subtractor;
 document.getElementById('zero').innerHTML = (Math.round(Setter * 100) /
100).toString();
}
```

Calculates total tank volume from measurements ($V = \pi r^2 h$) divided by 1000 to get litres, then updates text labels along the side of the tank evenly to two decimal places.

JS Loop:

```
temp = (analogIn(tempsensor) - 21) * 0.625;
document.getElementById("temperature").innerHTML = temp + "°C";
if (temp > 45) {
   turnOn(fan);
   document.getElementById("fanStatus").innerHTML = "Fan is ON";
   document.getElementById("fanStatus").setAttribute("style", "color:green;")
} else {
   turnOff(fan);
   document.getElementById("fanStatus").innerHTML = "Fan is OFF";
   document.getElementById("fanStatus").setAttribute("style", "color:red;")
}
if (Activate == 1) {
   sensor = (analogIn(UltrasonicSensor) - 105) * 2.652;
   if (sensor > h) {
      sensor = h;
   }
   output = Math.round((sensor) * (255 / h));
   if (output > 255) {
      output = 255;
   }
   if (output < 0) {
      output = 0;
   }
   StoredVals[i] = output;
   i++;
   if (i == 500) {
      i = 0;
      speakAgain = 1;
   }
   average = Math.round(arrAvg(StoredVals));
   if (average > 255) {
      average = 255;
   }
   if (average < 0) {
```

Check to see if button is pressed.

Gets sensor value from other board and caps it at the tank height.

Converts sensor value to a number between 0 and 255 (the tank graphic's height in pixels)

Adds output to the next spot (i) in the averaging list and resets i if it has reached 500.

Averages current list and ensures it is within the bounds of the graphic (just in case!)

Gets input from temperature sensor, converts to Celsius, and if it is too hot (>45°C) turns the fan on and updates the text to say "Fan is ON" in green.

```
    average = 0;
  }
  l1.setAttribute("y1", (average).toString());
  l1.setAttribute("y2", (average).toString());
  r1.setAttribute("y", (average).toString());
  r1.setAttribute("height", (255 - average).toString());
  WaterVol = Math.round((255 - average) * (TotalVol / 255) * 100) / 100;
  document.getElementById("value").innerHTML = WaterVol + "L of water";
  document.getElementById("alerts").innerHTML = "";
  speechSynthesis.cancel();
  if (average > 200 && speakAgain == 1) {
    document.getElementById("alerts").innerHTML = "Water is running low!";
    window.speechSynthesis.speak(msg1);
    speakAgain = 0;
    await mSec(2200);
  }
  if (average < 40 && speakAgain == 1) {
    document.getElementById("alerts").innerHTML = "Tank is almost full!";
    window.speechSynthesis.speak(msg2);
    speakAgain = 0;
    await mSec(2200);
  }
}
```

Sets height of water on graphic to the water level, then displays the exact volume of water

Display any alerts that have been triggered. (speakAgain is to ensure the TTS only happens every time the list resets to give the program time to catch up and so the program does not continually repeat itself.)



await mSec() is a custom function that is part of runlinc and that waits for the given amount of milliseconds.
analogIn() is used to get the input from a pin on the STEMSEL eLabtronics board by its name.

# Acknowledgement of External Help

The physical construction of the RTM (the tank, control panel, and parts of the main box) were constructed with the help of my older brother Nathan (27 years old, works as an engineer). He showed me how the many tools in his shed work (drill, router, electrical wire crimper and so on) and helped me use them to create the parts. The design, idea, wiring setup and code were all of my own creation.

# References:

I wrote the code for this a year ago (around June 2019) and so did not document absolutely every website I used, however the two main websites I used were w3schools (for learning new concepts) and stackoverflow (for debugging).

Stack Overflow. 2020. Stack Overflow - Where Developers Learn, Share, & Build Careers. [online] Available at: <https://stackoverflow.com/> [Accessed 14 July 2020].

W3schools.com. 2020. W3schools Online Web Tutorials. [online] Available at: <https://www.w3schools.com/> [Accessed 14 July 2020].

As for the research on the HC-SR04 ultrasonic sensor, I used these online tutorials to help me figure out its workings.

Last Minute Engineers. n.d. How HC-SR04 Ultrasonic Sensor Works & How To Interface It With Arduino. [online] Available at: <https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/> [Accessed 14 July 2020].

Santos, R., 2020. Complete Guide For Ultrasonic Sensor HC-SR04 With Arduino | Random Nerd Tutorials. [online] Random Nerd Tutorials. Available at: <https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/> [Accessed 14 July 2020].