5. PROGRAM A GREENHOUSE SENSE AND CONTROL SYSTEM

Project Overview

Students use an EcoChamber and a //control.Node along with all Greenhouse inputs (temperature, humidity, brightness, soil moisture) and outputs (USB Fan, USB Pump, Grow Light) to incorporate and modify existing Blockly code from previous investigations and use at least 2 sense-and-control approaches to program an autonomous Greenhouse.

Time Requirements

Teacher Preparation: 10 minutes

Student Project: 1 or more 45-minute class periods

Goals

- Incorporate existing code into a new program.
- Design a program that combines a variety of control structures to autonomously maintain a greenhouse.

Materials and Equipment

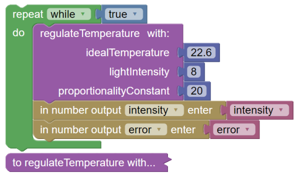
- Data collection system
- //control.Node
- Grow Light with included cables and USB power supply
- Greenhouse Sensor
- Greenhouse Sensor Module with cable and stopper
- Soil moisture probe
- USB Pump
- USB Fan
- Power Output Module with cable
- EcoChamber with lid and stoppers
- Assembled watering system*
- Potted plant, ~4" x ~4"
- Shallow dish
- Zip seal sandwich bag
- Ice

*Includes all Greenhouse Accessory Kit components (tubing, connectors, drip irrigation heads, hook-and-loop fasteners, and a #5 one-hole stopper), reservoir filled with tap water for USB pump, and materials to secure drip heads to the plant pot such as several strong rubber bands, zip-ties, and binder clips. Systems that require increased humidity may include a 2" x 2" sponge and bottle caps, however, once the plant is added to the Greenhouse, the setup may be modified as humidity conditions will change.

Teacher Tips

- For best results, fully charge wireless devices before starting the investigation.
- For long-term investigations, it is recommended to connect the //control.Node to continuous USB power.
- For Blockly help, enter code-related search terms in PASCO's online Blockly reference guide at **help.pasco.com/sparkvue**. The guide is also accessible from the **Help** option in SPARKvue's main menu ☰ while data collection is stopped.
- For helpful Greenhouse videos, visit PASCO's Greenhouse Sense and Control Kit video library (**click here** or scan the QR code).

- If your students plan on running the greenhouse long-term, code must be uploaded as directed in Part 2 to avoid data overload. The sample rate is 1 Hz, or 1 sample per second. When SPARKvue is run long-term with a fast sample rate plus several sensor measurements, a large amount of data will be recorded. This large data set translates to a large file size that could eventually overwhelm your system. However when code is uploaded to and is executed from the //control.Node, data is not being recorded, so large data sets are not an issue.
- For ease of access, consider setting up multiple-outlet power strips within easy reach for students. Plug the USB power supply into the power strip and connect the USB cable ahead of time so students can easily connect the Grow Light to the USB cable.
- Depending on your students' experience level, you may wish to plug the USB cable and flat Power Out cable into the Greenhouse Light for students ahead of time.
- This investigation requires background knowledge of Blockly variables, loops, functions, accessing Greenhouse Sensor measurements, operating the Power Output Module, creating numeric and text outputs, working with the USB fan, sleep delays, and creating data displays. To establish prior knowledge, students should complete the coding investigations titled *Program a Sunny Day for Plants*, *Code a Cooling Breeze for a Greenhouse*, *Program Perfectly Timed Rain*, and *Optimize Water Movement* before performing this investigation. The investigation titled *Optimize Water Movement* helps students determine ideal temperature, humidity, brightness, and soil moisture criteria for their plant.
- Optional (refer to sample code shown): Reinforce students' mathematical and visual ideas of what a constant is, and what it means when two variables have a *proportional relationship*. While students are working with the *Regulate Temperature* function in the Research section, have them create numeric outputs called *intensity* and *error*. Set each output equal to the *intensity* and *error* variables from the function. Create a graph of *intensity* versus *error*. Find the **Properties** ⚙ icon in the **Graph Tools** menu below the graph to turn **Show Connected Lines** off. Collect a few minutes of data. Add a linear fit 📈 to the results to get the slope *m* and y-intercept *b*. Slope should be about equal to the proportionality constant, and the y-intercept should be about equal the ideal intensity (as long as the ideal temperature is set 0.2 °C above ambient temperature as directed in the student instructions).



```
repeat while true
do  regulateTemperature with:
              idealTemperature 22.6
                lightIntensity 8
      proportionalityConstant 20
    in number output intensity enter intensity
    in number output error enter error

to regulateTemperature with...
```

- Optional, if the entire class is using the same type of plant of comparable initial size and health: After each student group creates their own autonomous code, have the class create one code together. Once the code is finalized, assign different student groups different light conditions: control with equal amounts of red and blue intensity; only red light; only blue light; and other red:blue light proportions. Students should only modify the Grow Light intensity settings in the class-designed code. Have the class monitor the effects on the plants over several weeks.

Safety

Follow these important safety precautions in addition to your regular classroom procedures:

- Keep water away from sensor boxes, electrical plugs, and exposed electronic boards.
- Don't allow exposed electronic boards to contact a metallic or conductive surface.

CAUTION:

- *Don't look directly at the LEDs.*
- *Don't touch the LEDs.*

Prototype

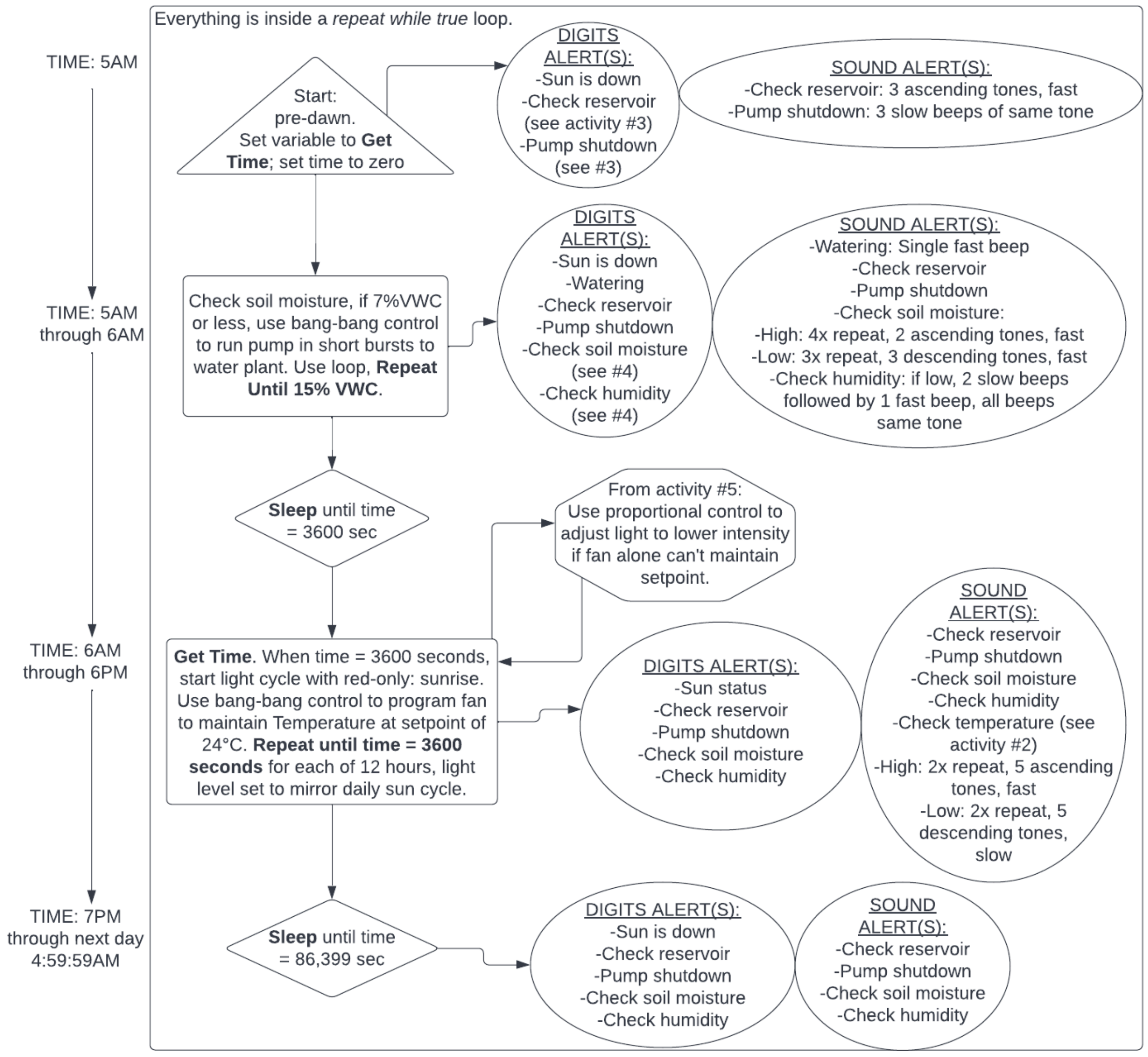
Part 4: Plan Your Code

On a separate paper, sketch a flow chart that shows how you plan to modify and combine existing code to create a single program that:

The flow chart should include the approximate time of day each event occurs as well as ideas of the types of blocks you will use. The flow chart does not need to show all lines of code; its purpose is to help you plan and think about the things that need to happen in a logical order with the correct kinds of variables, loops, sleep delays, etc. needed for a successful program. For example, a flow chart step might say "sleep for an hour then use a *for each item* loop with a list to gradually turn on the light" instead of listing the many blocks that would be required to complete the task.

Student plans should account for time as each event is identified; alerts associated with each event should also be identified and the audio signal for each event should be defined. Like the example shown, details will be missing as students discover unanticipated modifications they will need to make for the program to work on a 24-hour basis.

24-Hour Greenhouse Program Plan (Pothos Plant)


Everything is inside a *repeat while true* loop.
TIME: 5AM
Start: pre-dawn. Set variable to **Get Time**; set time to zero
DIGITS ALERT(S): -Sun is down -Check reservoir (see activity #3) -Pump shutdown (see #3)
SOUND ALERT(S): -Check reservoir: 3 ascending tones, fast -Pump shutdown: 3 slow beeps of same tone
TIME: 5AM through 6AM
Check soil moisture, if 7%VWC or less, use bang-bang control to run pump in short bursts to water plant. Use loop, **Repeat Until 15% VWC**.
DIGITS ALERT(S): -Sun is down -Watering -Check reservoir -Pump shutdown -Check soil moisture (see #4) -Check humidity (see #4)
SOUND ALERT(S): -Watering: Single fast beep -Check reservoir -Pump shutdown -Check soil moisture: -High: 4x repeat, 2 ascending tones, fast -Low: 3x repeat, 3 descending tones, fast -Check humidity: if low, 2 slow beeps followed by 1 fast beep, all beeps same tone
Sleep until time = 3600 sec
From activity #5: Use proportional control to adjust light to lower intensity if fan alone can't maintain setpoint.
TIME: 6AM through 6PM
Get Time. When time = 3600 seconds, start light cycle with red-only: sunrise. Use bang-bang control to program fan to maintain Temperature at setpoint of 24°C. **Repeat until time = 3600 seconds** for each of 12 hours, light level set to mirror daily sun cycle.
DIGITS ALERT(S): -Sun status -Check reservoir -Pump shutdown -Check soil moisture -Check humidity
SOUND ALERT(S): -Check reservoir -Pump shutdown -Check soil moisture -Check humidity -Check temperature (see activity #2) -High: 2x repeat, 5 ascending tones, fast -Low: 2x repeat, 5 descending tones, slow
TIME: 7PM through next day 4:59:59AM
Sleep until time = 86,399 sec
DIGITS ALERT(S): -Sun is down -Check reservoir -Pump shutdown -Check soil moisture -Check humidity
SOUND ALERT(S): -Check reservoir -Pump shutdown -Check soil moisture -Check humidity

Test

1. Summarize the events that occur in your 24-hour program, and explain how your program was written to ensure the correct timing over a 24-hour period.

   The main code is the repeating 24-hour loop, and inside that code, the day begins at 5AM before sunrise when soil moisture is checked and the plant is watered if needed. Watering can only occur at this time and alerts will sound and display if the water reservoir gets too low; the pump can also be disabled if water is too low. Then sunrise begins at 6AM. Light starts red only then blue is gradually added as intensity gradually increases until the peak at noon, then intensity decreases until sunset at 6PM. While the light is on, proportional control is used to reduce light intensity if necessary to maintain temperature setpoint. Reservoir level, humidity, temperature, and soil moisture are checked once per hour and alerts are sounded/displayed if needed. Then the system goes dark until 24 hours have passed, and the cycle begins again at the same time on the next day.

2. What were the top 2 challenges you faced when combining multiple input and output events into a single program?

   For most of the investigations, a *repeat while true* worked but for the 24-hour cycle, you couldn't use that loop for individual functions because the program would get stuck in a forever-repeating loop and never make it to the next function. You had to think about the type of loop, if any, to use for every function or you had to add a conditional statement to make sure an alert from the previous day made it into the next day. Also figuring out how to sequence one event to the next and the sleep time between each was a challenge. This was hard to think about because we were not sure when one 24-hour cycle stopped and the next began, until we thought to create a "day counter" variable and add it to a Digits display.

3. What advice would you give your peers on how to be successful when combining multiple input and output events into a single program?

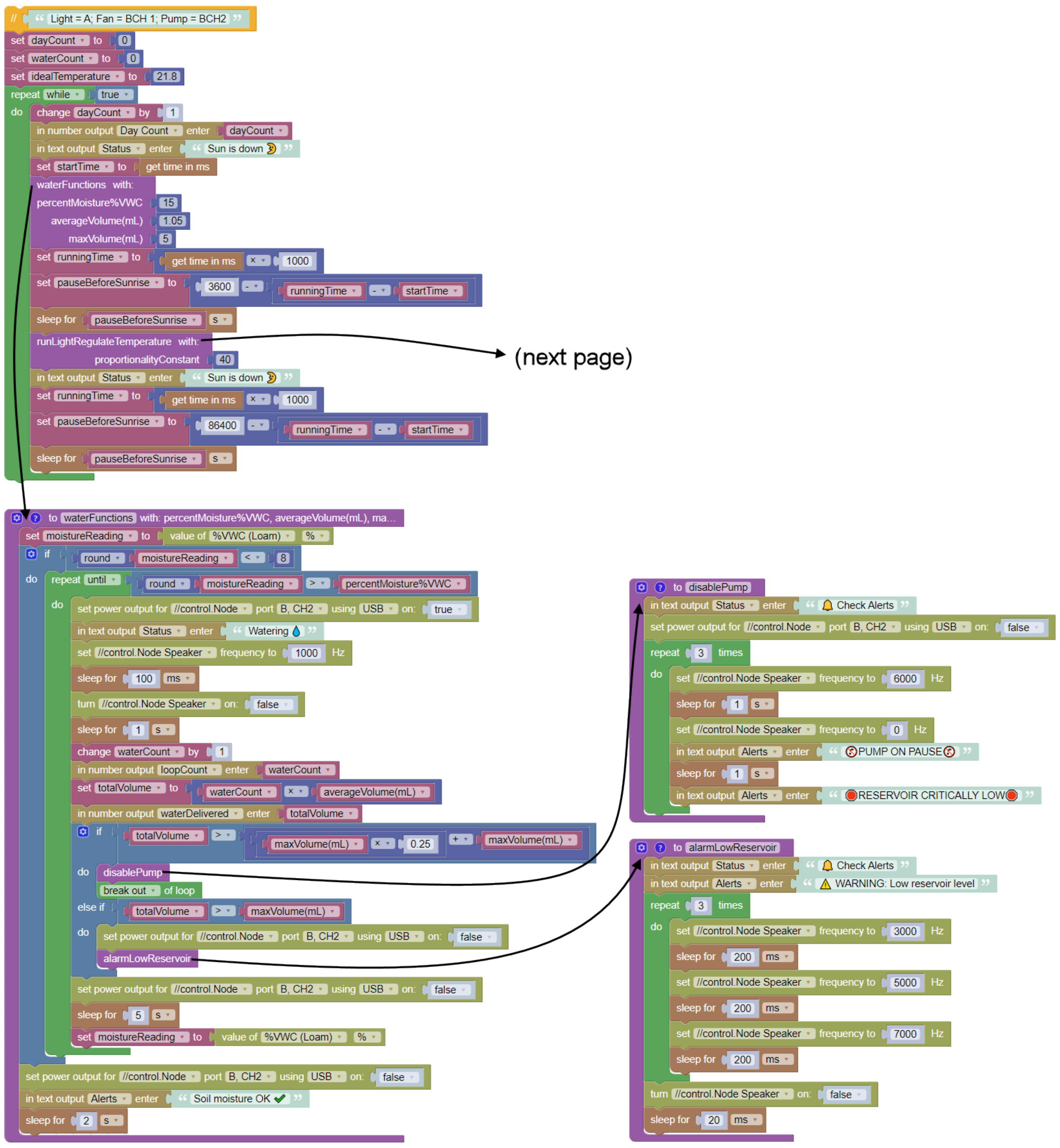
   You need to break down all the pieces of the program into separate functions that a spread out over time with sleep blocks. Think very carefully about the order in which things have to happen. Take the troubleshooting advice like use table data displays for variables and measurements to make sure what's actually going on is what you think is going on, and also code a little bit, then test - testing often saves time (and frustration!) in the long run. Finally, if someone has already done the work to write code for something you need, review their code to see if you can use some or all of it. As long as you understand how the code works, don't reinvent the wheel if you don't need to, but do look for opportunities to improve it.

4. Explain your approach to using proportional control in your 24-hour program. Describe at least one advantage this type of control has over bang-bang control.

   I merged code from the *Regulate Temperature* function with an array (built from lists) and a *for each item* loop to maximize light only as much as was appropriate for the time of day, so light couldn't increase above a maximum, but could be reduced gradually if the fan alone could not keep the Greenhouse at setpoint. One benefit of proportional control is you can spend more time closer to the ideal temperature and light values compared to bang-bang control.

Sample code, part 1 begins on the next page.

Sample code, Part 1 (continues on next page):



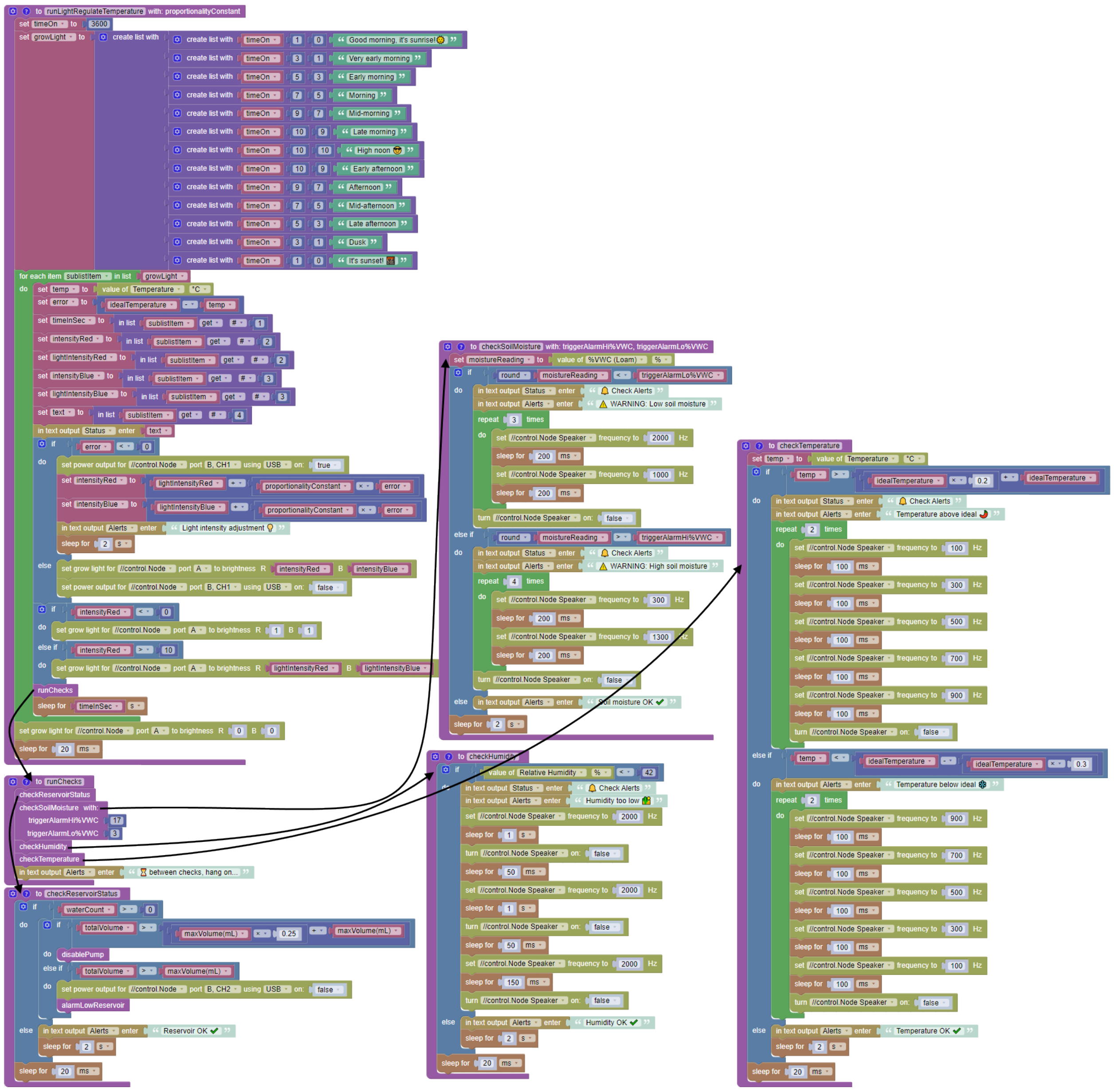
```
// "Light = A; Fan = BCH 1; Pump = BCH2"
set dayCount to 0
set waterCount to 0
set idealTemperature to 21.8
repeat while true
do  change dayCount by 1
    in number output Day Count enter dayCount
    in text output Status enter "Sun is down 🌛"
    set startTime to get time in ms
    waterFunctions with:
        percentMoisture%VWC 15
        averageVolume(mL) 1.05
        maxVolume(mL) 5
    set runningTime to get time in ms × 1000
    set pauseBeforeSunrise to 3600 - (runningTime - startTime)
    sleep for pauseBeforeSunrise s
    runLightRegulateTemperature with:
        proportionalityConstant 40
    in text output Status enter "Sun is down 🌛"
    set runningTime to get time in ms × 1000
    set pauseBeforeSunrise to 86400 - (runningTime - startTime)
    sleep for pauseBeforeSunrise s
```

```
to waterFunctions with: percentMoisture%VWC, averageVolume(mL), ma...
    set moistureReading to value of %VWC (Loam) %
    if round moistureReading < 8
    do  repeat until round moistureReading > percentMoisture%VWC
        do  set power output for //control.Node port B, CH2 using USB on: true
            in text output Status enter "Watering 💧"
            set //control.Node Speaker frequency to 1000 Hz
            sleep for 100 ms
            turn //control.Node Speaker on: false
            sleep for 1 s
            change waterCount by 1
            in number output loopCount enter waterCount
            set totalVolume to waterCount × averageVolume(mL)
            in number output waterDelivered enter totalVolume
            if totalVolume > (maxVolume(mL) × 0.25) + maxVolume(mL)
            do  disablePump
                break out of loop
            else if totalVolume > maxVolume(mL)
            do  set power output for //control.Node port B, CH2 using USB on: false
                alarmLowReservoir
            set power output for //control.Node port B, CH2 using USB on: false
            sleep for 5 s
            set moistureReading to value of %VWC (Loam) %
    set power output for //control.Node port B, CH2 using USB on: false
    in text output Alerts enter "Soil moisture OK ✔"
    sleep for 2 s
```

```
to disablePump
    in text output Status enter "🔔 Check Alerts"
    set power output for //control.Node port B, CH2 using USB on: false
    repeat 3 times
    do  set //control.Node Speaker frequency to 6000 Hz
        sleep for 1 s
        set //control.Node Speaker frequency to 0 Hz
        in text output Alerts enter "☹PUMP ON PAUSE☹"
        sleep for 1 s
        in text output Alerts enter "🔴RESERVOIR CRITICALLY LOW🔴"
```

```
to alarmLowReservoir
    in text output Status enter "🔔 Check Alerts"
    in text output Alerts enter "⚠ WARNING: Low reservoir level"
    repeat 3 times
    do  set //control.Node Speaker frequency to 3000 Hz
        sleep for 200 ms
        set //control.Node Speaker frequency to 5000 Hz
        sleep for 200 ms
        set //control.Node Speaker frequency to 7000 Hz
        sleep for 200 ms
    turn //control.Node Speaker on: false
    sleep for 20 ms
```

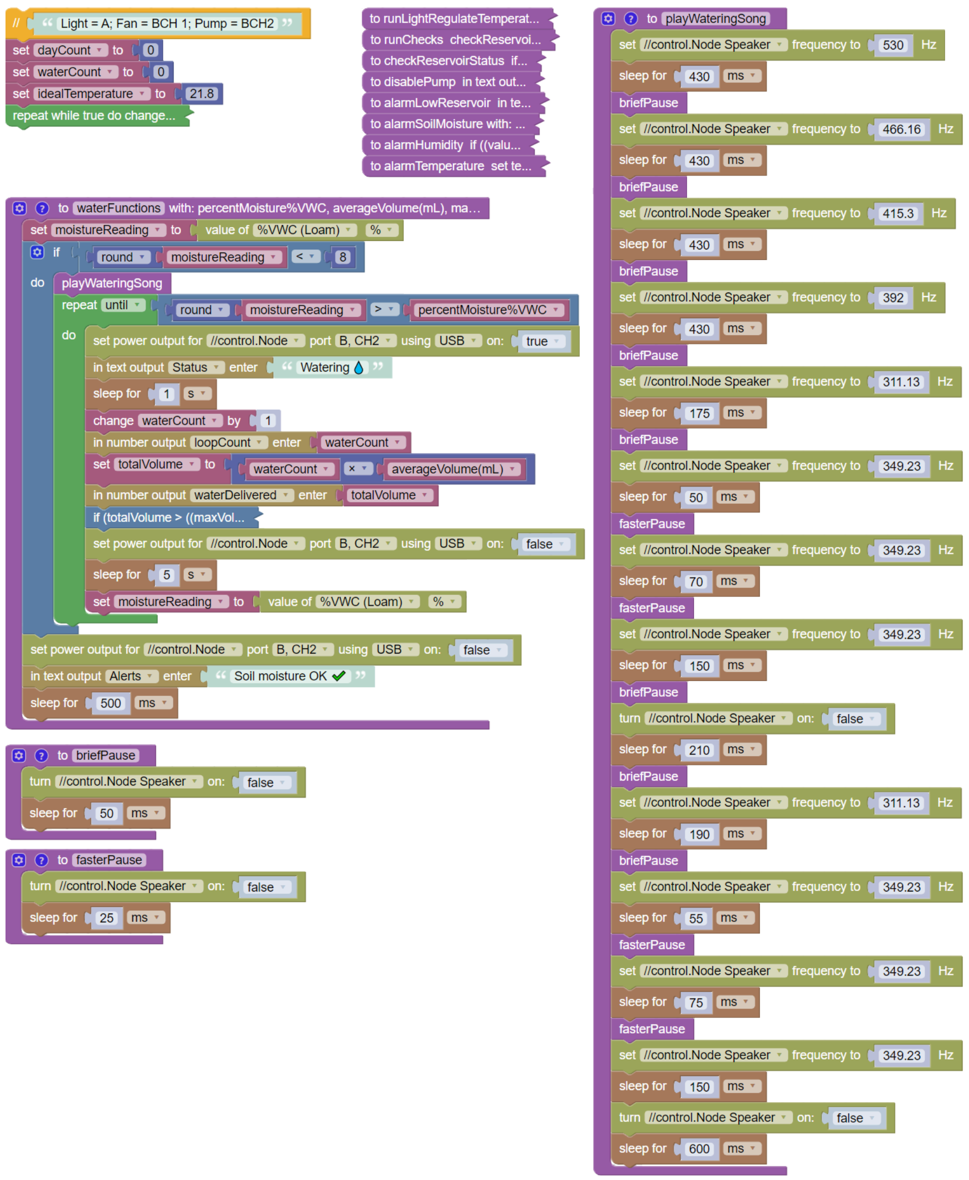
Sample code, Part 2 (continued from previous page):



```
to runLightRegulateTemperature with: proportionalityConstant
  set timeOn to 3600
  set growLight to create list with
      create list with timeOn 1 0 "Good morning, it's sunrise!🌞"
      create list with timeOn 3 1 "Very early morning"
      create list with timeOn 5 3 "Early morning"
      create list with timeOn 7 5 "Morning"
      create list with timeOn 9 7 "Mid-morning"
      create list with timeOn 10 9 "Late morning"
      create list with timeOn 10 10 "High noon 😎"
      create list with timeOn 10 9 "Early afternoon"
      create list with timeOn 9 7 "Afternoon"
      create list with timeOn 7 5 "Mid-afternoon"
      create list with timeOn 5 3 "Late afternoon"
      create list with timeOn 3 1 "Dusk"
      create list with timeOn 1 0 "It's sunset! 🌇"
  for each item sublistItem in list growLight
  do  set temp to value of Temperature °C
      set error to idealTemperature - temp
      set timeInSec to in list sublistItem get # 1
      set intensityRed to in list sublistItem get # 2
      set lightIntensityRed to in list sublistItem get # 2
      set intensityBlue to in list sublistItem get # 3
      set lightIntensityBlue to in list sublistItem get # 3
      set text to in list sublistItem get # 4
      in text output Status enter text
      if error < 0
      do   set power output for //control.Node port B, CH1 using USB on: true
           set intensityRed to lightIntensityRed + proportionalityConstant × error
           set intensityBlue to lightIntensityBlue + proportionalityConstant × error
           in text output Alerts enter "Light intensity adjustment 💡"
           sleep for 2 s
      else set grow light for //control.Node port A to brightness R intensityRed B intensityBlue
           set power output for //control.Node port B, CH1 using USB on: false
      if intensityRed < 0
      do   set grow light for //control.Node port A to brightness R 1 B 1
      else if intensityRed > 10
      do   set grow light for //control.Node port A to brightness R lightIntensityRed B lightIntensityBlue
      runChecks
      sleep for timeInSec s
  set grow light for //control.Node port A to brightness R 0 B 0
  sleep for 20 ms

to runChecks
  checkReservoirStatus
  checkSoilMoisture with:
      triggerAlarmHi%VWC 17
      triggerAlarmLo%VWC 3
  checkHumidity
  checkTemperature
  in text output Alerts enter "⏳ between checks, hang on..."

to checkReservoirStatus
  if waterCount > 0
  do   if totalVolume > maxVolume(mL) × 0.25 + maxVolume(mL)
       do disablePump
       else if totalVolume > maxVolume(mL)
       do set power output for //control.Node port B, CH2 using USB on: false
          alarmLowReservoir
  else in text output Alerts enter "Reservoir OK ✔"
       sleep for 2 s
  sleep for 20 ms

to checkSoilMoisture with: triggerAlarmHi%VWC, triggerAlarmLo%VWC
  set moistureReading to value of %VWC (Loam) %
  if round moistureReading < triggerAlarmLo%VWC
  do   in text output Status enter "🔔 Check Alerts"
       in text output Alerts enter "⚠ WARNING: Low soil moisture"
       repeat 3 times
       do   set //control.Node Speaker frequency to 2000 Hz
            sleep for 200 ms
            set //control.Node Speaker frequency to 1000 Hz
            sleep for 200 ms
       turn //control.Node Speaker on: false
  else if round moistureReading > triggerAlarmHi%VWC
  do   in text output Status enter "🔔 Check Alerts"
       in text output Alerts enter "⚠ WARNING: High soil moisture"
       repeat 4 times
       do   set //control.Node Speaker frequency to 300 Hz
            sleep for 200 ms
            set //control.Node Speaker frequency to 1300 Hz
            sleep for 200 ms
       turn //control.Node Speaker on: false
  else in text output Alerts enter "Soil moisture OK ✔"
  sleep for 2 s

to checkHumidity
  if value of Relative Humidity % < 42
  do   in text output Status enter "🔔 Check Alerts"
       in text output Alerts enter "Humidity too low 🌵"
       set //control.Node Speaker frequency to 2000 Hz
       sleep for 1 s
       turn //control.Node Speaker on: false
       sleep for 50 ms
       set //control.Node Speaker frequency to 2000 Hz
       sleep for 1 s
       turn //control.Node Speaker on: false
       sleep for 50 ms
       set //control.Node Speaker frequency to 2000 Hz
       sleep for 150 ms
       turn //control.Node Speaker on: false
  else in text output Alerts enter "Humidity OK ✔"
       sleep for 2 s
  sleep for 20 ms

to checkTemperature
  set temp to value of Temperature °C
  if temp > idealTemperature × 0.2 + idealTemperature
  do   in text output Status enter "🔔 Check Alerts"
       in text output Alerts enter "Temperature above ideal 🌶"
       repeat 2 times
       do   set //control.Node Speaker frequency to 100 Hz
            sleep for 100 ms
            set //control.Node Speaker frequency to 300 Hz
            sleep for 100 ms
            set //control.Node Speaker frequency to 500 Hz
            sleep for 100 ms
            set //control.Node Speaker frequency to 700 Hz
            sleep for 100 ms
            set //control.Node Speaker frequency to 900 Hz
            sleep for 100 ms
            turn //control.Node Speaker on: false
  else if temp < idealTemperature - idealTemperature × 0.3
  do   in text output Alerts enter "Temperature below ideal ❄"
       repeat 2 times
       do   set //control.Node Speaker frequency to 900 Hz
            sleep for 100 ms
            set //control.Node Speaker frequency to 700 Hz
            sleep for 100 ms
            set //control.Node Speaker frequency to 500 Hz
            sleep for 100 ms
            set //control.Node Speaker frequency to 300 Hz
            sleep for 100 ms
            set //control.Node Speaker frequency to 100 Hz
            sleep for 100 ms
            turn //control.Node Speaker on: false
  else in text output Alerts enter "Temperature OK ✔"
       sleep for 2 s
  sleep for 20 ms
```

Improve

The chorus from the Bee Gees' *Stayin' Alive* will play while watering instead of the single beep. Unchanged code has been collapsed (sample code on the next page).



```
// "Light = A; Fan = BCH 1; Pump = BCH2"
set dayCount to 0
set waterCount to 0
set idealTemperature to 21.8
repeat while true do change...

to runLightRegulateTemperat...
to runChecks checkReservoi...
to checkReservoirStatus if...
to disablePump in text out...
to alarmLowReservoir in te...
to alarmSoilMoisture with: ...
to alarmHumidity if ((valu...
to alarmTemperature set te...

to waterFunctions with: percentMoisture%VWC, averageVolume(mL), ma...
  set moistureReading to value of %VWC (Loam) %
  if round moistureReading < 8
  do playWateringSong
     repeat until round moistureReading > percentMoisture%VWC
     do set power output for //control.Node port B, CH2 using USB on: true
        in text output Status enter "Watering 💧"
        sleep for 1 s
        change waterCount by 1
        in number output loopCount enter waterCount
        set totalVolume to waterCount × averageVolume(mL)
        in number output waterDelivered enter totalVolume
        if (totalVolume > ((maxVol...
        set power output for //control.Node port B, CH2 using USB on: false
        sleep for 5 s
        set moistureReading to value of %VWC (Loam) %
  set power output for //control.Node port B, CH2 using USB on: false
  in text output Alerts enter "Soil moisture OK ✔"
  sleep for 500 ms

to briefPause
  turn //control.Node Speaker on: false
  sleep for 50 ms

to fasterPause
  turn //control.Node Speaker on: false
  sleep for 25 ms

to playWateringSong
  set //control.Node Speaker frequency to 530 Hz
  sleep for 430 ms
  briefPause
  set //control.Node Speaker frequency to 466.16 Hz
  sleep for 430 ms
  briefPause
  set //control.Node Speaker frequency to 415.3 Hz
  sleep for 430 ms
  briefPause
  set //control.Node Speaker frequency to 392 Hz
  sleep for 430 ms
  briefPause
  set //control.Node Speaker frequency to 311.13 Hz
  sleep for 175 ms
  briefPause
  set //control.Node Speaker frequency to 349.23 Hz
  sleep for 50 ms
  fasterPause
  set //control.Node Speaker frequency to 349.23 Hz
  sleep for 70 ms
  fasterPause
  set //control.Node Speaker frequency to 349.23 Hz
  sleep for 150 ms
  briefPause
  turn //control.Node Speaker on: false
  sleep for 210 ms
  briefPause
  set //control.Node Speaker frequency to 311.13 Hz
  sleep for 190 ms
  briefPause
  set //control.Node Speaker frequency to 349.23 Hz
  sleep for 55 ms
  fasterPause
  set //control.Node Speaker frequency to 349.23 Hz
  sleep for 75 ms
  fasterPause
  set //control.Node Speaker frequency to 349.23 Hz
  sleep for 150 ms
  turn //control.Node Speaker on: false
  sleep for 600 ms
```

Technical Support

Need more help? Our knowledgeable and friendly Technical Support staff is ready to provide assistance with this or any other PASCO product.

Phone (USA)	1-800-772-8700 (Option 4)
Phone (International)	+1 916 462 8384
Online	**pasco.com/support**