Growing Beyond Earth Control Box

Designed and built by Fairchild Tropical Botanic Garden
Limited use prototype version 0.61, October 20, 2021

Essential Information

- This is an experimental Growing Beyond Earth kit released only to South Florida high schools for the 2021-22 school year.
- It includes a 24-volt power supply, a switch, and a Control Box that is attached to the LED panel atop the growth chamber (Figure 1).
- The power supply should be plugged directly into a wall outlet with no timer.
- For comparison, Figure 2 shows the older version of the Growing Beyond Earth kit including an outlet timer, fan power supply, and LED controller. Those components are now unnecessary because their functions are handled by the Control Box.
- The Control Box operates the LED lights and fan by running a computer program. The Control Box comes pre-loaded with a program that has the proper timing and settings for the 2021 radish experiment.
- The white LED on the Control Box pulses on and off to indicate that the computer program is running properly.
- The Control Box does not need to be connected to a computer to run.
- If you choose, you can connect your computer to the Control Box with the included USB cable. Then, you can edit the code and modify the light and fan settings.

Resources for programming the Control Box

- Repository of program files and information on the Control Box: https://github.com/Growing-Beyond-Earth/control-box
- Thonny software for editing code on the Control Box: https://thonny.org/
- Book on coding for the Raspberry Pi Pico, the “brain” of the Control Box: https://hackspace.raspberrypi.com/books/micropython-pico
- Online tutorial on connecting to and programming the Raspberry Pi Pico https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico

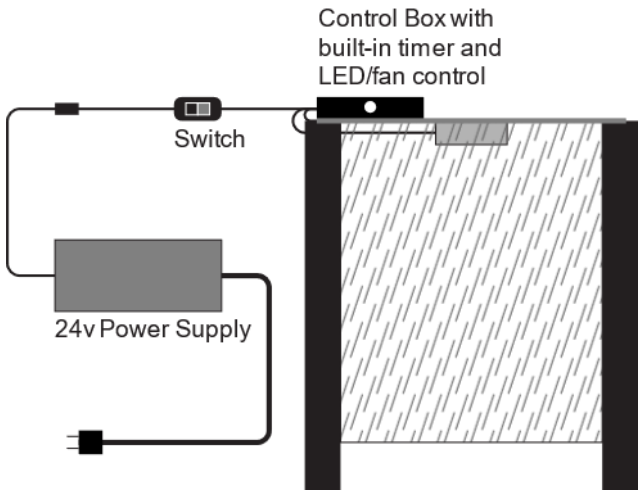

Figure 1. New (2021-2022) Growing Beyond Earth kit with Control Box

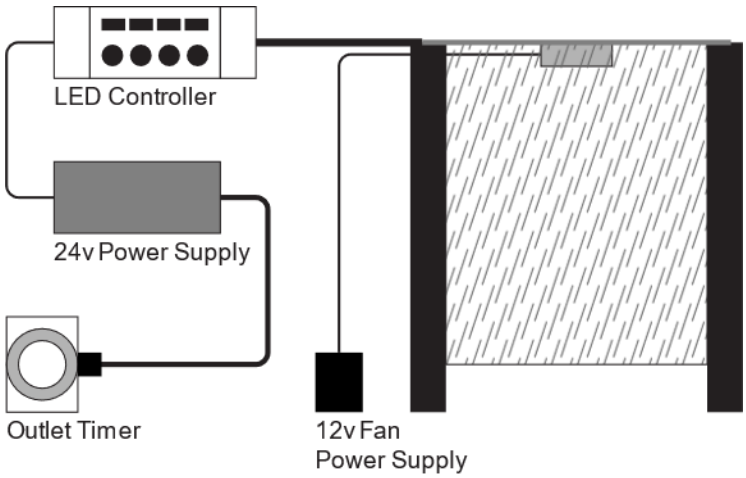

Figure 2. Old (2019-2021) Growing Beyond Earth kit

Background

Fairchild Tropical Botanic Garden developed Growing Beyond Earth (GBE) to get students involved with NASA's research on plants. When the program began in 2015, Fairchild distributed a kit of off-the-shelf parts that formed a plant habitat like those aboard the International Space Station (ISS). Although that first kit worked well in classrooms and yielded useful data, it had several limitations. The LEDs were too bright and hot, the system lacked ventilation, and it was difficult to water the plants evenly. Over the years we addressed those limitations by ditching the off-the-shelf parts and developing new components from scratch. The GBE Control Box is our latest new component, now ready for testing in a limited number of classrooms. We created it to provide more consistent environmental control while giving students access to the underlying technology.

Sitting atop the growth chamber, the GBE Control Box runs the LED lighting and fan. It works automatically with built-in programming that specifies the on/off timing of the lights, the brightness of each LED channel, and the fan speed. Although it can be programmed by a computer, it does not need to be connected to a computer to run. The system is easier to set up and operate than previous, manually controlled versions of the GBE kit — just plug it in and it works — but it can also be more powerful. Students can develop their own computer programs to control the system and run sophisticated experiments.

The GBE Control Box is built around a Raspberry Pi Pico microcontroller, a tiny single-board computer that runs simple programs. The Pico can be programmed in Python or C/C++ using development software that runs on Mac, PC, and Linux. Many online resources are available to help teachers and students get started with coding on the Pico.

As soon as the GBE Control Box is connected to power, the Raspberry Pi Pico begins running code. In our default setup, the Pico runs a Python program that turns the lights on in the morning and off twelve hours later, reading the time from a built-in clock. It sets the LEDs to medium brightness with a mixture of red, green, blue, and white. The program runs the fan at full power during the day and half-power overnight.

The default program works well for many kinds of plants, but students can modify it to create different conditions. With simple changes to the code, the on/off times, intensity, and color balance of the lights can be adjusted. More advanced coding can manipulate the lights at specific stages of the experiment and at different times during the day.

With some tinkering and programming, the Pico can do more than just control the lights and fan. It can connect to environmental sensors, opening the door to programs that will log data and respond to changes in temperature, humidity, and soil moisture. Other electronic devices, including small water pumps, can be connected as well. Students may be able to design a system, built around the GBE Control Box, that takes care of plants automatically.

We set up the Raspberry Pi Pico with Micropython, a lightweight version of the Python programming language optimized for the low-powered CPUs and limited memory of microcontrollers. There are excellent online resources available to help you learn to code in Micropython.

Inside the GBE Control Box

The GBE Control Box is built from many small circuit boards connected within a 3D printed case. Once you learn how it all works, you are welcome to modify it or build your own from scratch, adding whatever capabilities you want. If you modify the system, please let us know what you try and what you learn. Below, we provide an explanation of the major components of the Control Box. Please see the attached wiring diagrams to see how everything is connected.

Raspberry Pi Pico

Serving as the brain of the Control Box, the Pico runs code that operates the lights, fan, and any other attached devices. You may be familiar with other models of Raspberry Pi that are full-fledged computers running Linux, but the Pico is a comparatively lightweight system. Like other microcontrollers, its software interacts with devices attached to general purpose input-output (GPIO) connectors. The Pico has 26 GPIO connectors along with controls for power and ground and other miscellaneous features (e.g., system control and debugging). The pin assignments in the Control Box are shown in Figure 3.

Power

Electrical power is fed from a 24-volt wall adapter into a port on the back of the GBE Control Box. We include an inline switch that makes it easier to turn the power on and off. The LED light panel operates at 24 volts, but all the other circuitry runs on 12 volts or 3.3 volts. To have three separate power inputs providing 24, 12, and 3.3 volts, the Control Box includes two step-down converters that drop the 24-volt input to the lower voltages. The Raspberry Pi Pico and the I2C bus (explained below) run at 3.3 volts while the fan and external auxiliary port run at 12 volts.

LED light control

The GBE light panel is a custom-built aluminum circuit board designed by Fairchild and the Horticultural Lighting Group. It has 157 individual LED lights (40 red, 14 blue, 7 green, and 96 white). The mixture of these colors, called the “light recipe,” can be adjusted by dimming the LED color channels independently. The ratio of red, blue, and green LEDs was designed to match the Veggie growth chamber on ISS.

The brightness of each LED channel is controlled through a method called pulse width modulation (PWM). The LEDs are pulsed on and off thousands of times per second, fast enough to make the pulsing is imperceptible to our eyes. Dimming is achieved by adjusting the “width” of each of these pulses, the amount of time the lights are kept on per on/off cycle. The pulse width, called the *duty cycle*, can be zero (lights off) to 100% (lights at maximum brightness) and any value in between. Rather than using percentages, our built-in Micropython code expresses the duty cycle as an integer from zero to 255.

The Raspberry Pi Pico generates pulses that control the LEDs, with the specific frequency and duty cycle set within the code. The LED lights vibrate slightly when they pulse on and off, and that vibration can be audible at certain PWM frequencies. Our Micropython code sets the frequency to 20,000 Hertz to keep the sound at a higher pitch than our ears can hear. The Pico runs at 3.3 volts and does not carry enough power to light the LED panel on its own, so its pulse signals need to be amplified by additional circuitry.

MOSFETs (metal-oxide-semiconductor field-effect transistors) amplify the 3.3-volt PWM pulses

from the Raspberry Pi Pico to the full 24 volts needed to drive the LED lights. Four MOSFETs are connected to +24 volts (red wire) and ground (black wire), and each is connected to a separate PWM channel (red, green, blue, white) from the Pico. Five wires connect these MOSFETs to the LED light panel: one negative wire for each of the four LED channels plus a single positive wire (yellow).

Fan

Mounted in the center of the LED light panel, the fan circulates air through the growth chamber. It is a small, 12-volt fan designed to be used inside a computer. Just like a computer can regulate the speed of its fan, the Pico can control the growth chamber fan to increase or decrease air flow. The fan speed is regulated by PWM signals, so the code is similar to what we use to control the LED lights. By default, our code runs the fan at full speed while the lights are on and half speed while the lights are off.

I^2C bus 0, I^2C bus 1

Microcontrollers like the Pico are designed to communicate with many kinds of input and output devices. To be able to transmit and receive data, those devices and the Pico need a common communication protocol. I^2C, the *Inter-Integrated Circuit* protocol, was developed nearly 40 years ago to simplify communication among computer components and peripherals. Today, there are many low-cost I^2C devices that can be interconnected with one another and the Pico.

The Pico has two separate I^2C buses (communication channels), designated 0 and 1. Each bus can have many connected I^2C devices. We use bus 0 to connect the Pico to the real time clock and an electric current sensor inside the control box. Bus 1 is available for additional devices that can be connected via the 4-pin connector on the outside the control box.

Status LED

There is a single white LED on the outside of the Control Box that can be programmed to indicate the status of a running program. When our default code is running, the status LED pulses on and off.

12v auxiliary connector

A fifth MOSFET is set up as an electronic switch that can toggle a 12-volt accessory, like a water pump or auxiliary fan, on and off. This fifth MOSFET is connected via a yellow wire to a pin on the Pico that is configured to be either on or off. It is fed power from the 12-volt power converter, and its output is connected to a 2-pin auxiliary port on the back of the Control Box.

Programming Quick-Start Guide

1. Switch off the main power to your Control Box.
2. Use the included USB cable to connect your computer to the Control Box.
3. Load Thonny (downloaded from thonny.org) on your computer.
4. Hit the stop button in the Thonny toolbar—That will stop any running code so you can edit files on the Pico.
5. Switch on the main power to the control box. That will allow you to run programs and test code from within Thonny.
6. Choose File -> Open
7. Choose “Raspberry Pi Pico”
8. Open and run the **RTCMachineSet.py** file to synchronize the clock inside the Control Box with your computer’s clock.
9. Open the **main.py** file to see and edit the code that automatically runs each time the Control Box is powered up. This is the file that includes the basic LED and fan settings, shown in Figure 4.
10. Save the file or hit the run button when you are done editing the file. Hitting the run button saves the file before executing it.
11. Disconnect the computer.
12. Switch the main Control Box power off and back on again to execute the new code.

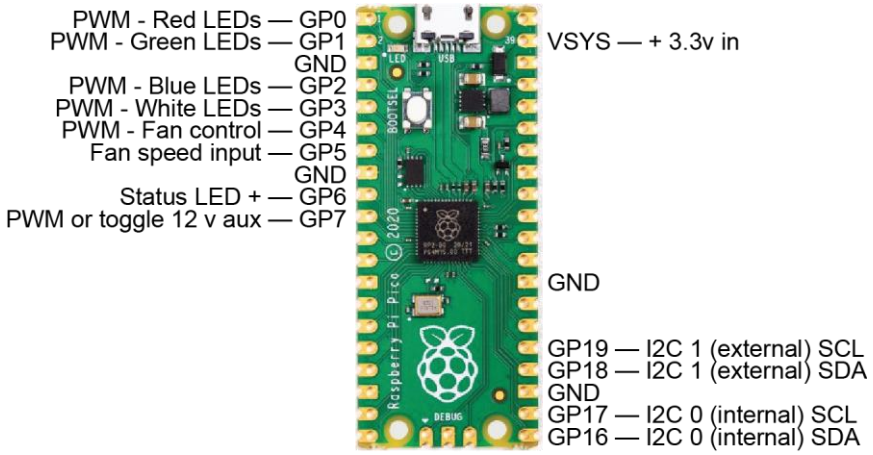

Figure 3. Raspberry Pi Pico pin assignments in the Growing Beyond Earth Control Box

Basic LED and fan settings

We made it easy for you to change basic settings within the Control Box without any coding knowledge. The main program running on the Control Box, called **main.py**, has a section near the top that specifies when the lights are turned on and off, the brightness of each LED channel, and the day and night power settings for the fan. To change these settings, load the **main.py** file in Thonny, look for the section shown in Figure 4, and edit the highlighted values.

```
# ----------------SETTINGS FOR LIGHTS AND FAN-------------------
# LIGHTS -- Time values are hh:mm strings based on a 24h clock;
#           brightness values are integers from 0 to 255 that set
#           the PWM duty cycle

lights_on_time = "07:00"
lights_off_time = "19:00"

red_brightness = 72
green_brightness = 60
blue_brightness = 44
white_brightness = 52


# FAN -- Fan power is an integer from 0 to 255 that sets the PWM
#        duty cycle

fan_power_when_lights_on = 255
fan_power_when_lights_off = 128

# -------------------------------------------------------------
```

Figure 4. Section of *main.py* where LED and fan settings are entered.