User Manual

SCUTTLE Power Management System

NexTec

Sponsored By
David Malawey

Authors

Daniel Zoch

Patrick Bowers

Hayden Bowen

Cameron Travis

User Manual

Overview

The purpose of this user manual is to provide instruction on how to operate and use the SCUTTLE Power Management System. This manual contains information covering topics which are unique to the S.P.M.S. and are NOT covered already in the MXET 300 course or accompanying provisions. The topics which are specifically and exclusively covered in this manual are listed below.

Quick Start Guide

Introduction

The Quick Start Guide is designed for users who are already provided with the following:
-A fully assembled S.P.M.S. Charging Station with accompanying software installed and running
-A SCUTTLE On Board module with wireless receiver pads and RFID tag properly attached
-A S.P.M.S. compatible SCUTTLE including a webcam with appropriate mounting attachments and supporting software properly installed including both the basic files for MXET 300 AND the files provided with the S.P.M.S. (A list of these files can be found in the "Software Overview" section of this manual)

In essence, this portion of the manual will allow you to quickly mount your S.O.B. in the proper position as well as your webcam and show you how to use the docking sequence function in your code so that you are instantly ready to autonomously dock and charge your SCUTTLE!

Setting Up The Charging Station

If your Charging Station is already assembled and the supporting software is properly installed, then the setup process for the Charging Station is very simple. Just plug the station into any U.S. Standard wall outlet and ensure the area around the Station described in Figure 1.0 is free of any obstructions.

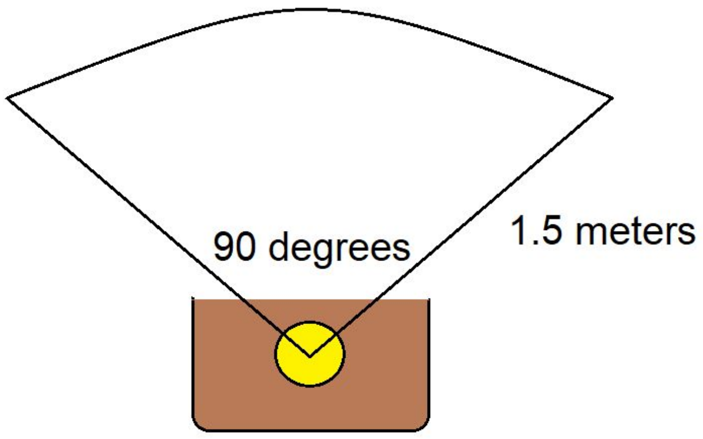


Figure 1.0: Charging Station Obstacle Free Zone

Once properly setup, the Station's Three Wireless base pads should have dimly lit red indicators on the sides and the MFRC522 RFID Reader should have a lit red LED. The complete assembly should mimic the image shown below in Figure 1.1.

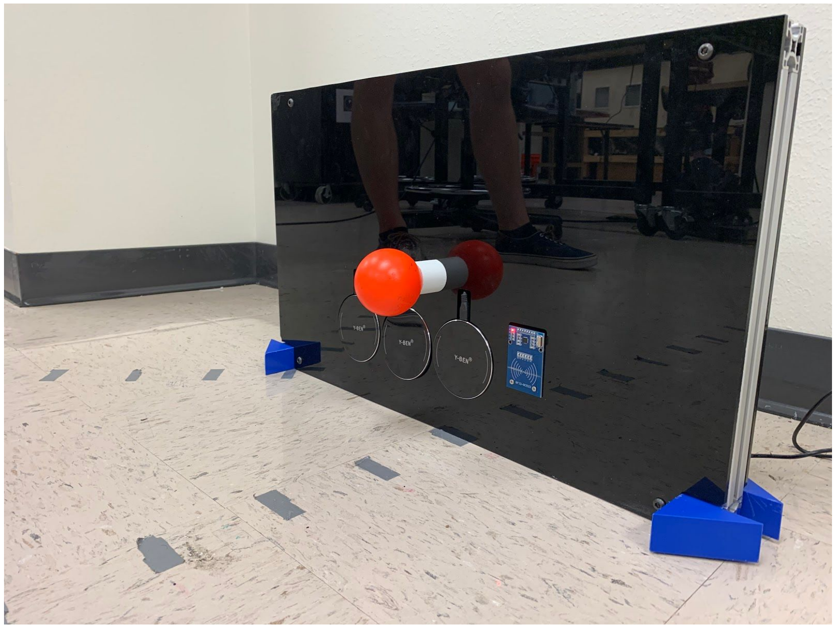


Figure 1.1: Charging Station Full Assembly

Setting Up The SCUTTLE On Board Module and Webcam

If your S.O.B. PCB is already assembled for you, the steps for installing it on your SCUTTLE are as follows:

1) Attach front panel to the front of the front rail of SCUTTLE. (The front panel should already have the wireless receiver pads and RFID reader attached to it. If not, refer to "The On-Board Module")
2) Attach S.O.B. PCB to the rear of the front rail of SCUTTLE. (The PCB should already have the mounting clips attached to it. If not, refer to "The On-Board Module")
3) Follow the wiring graphic shown below in Figure 1.2 and ensure the proper connections are made. (Your battery pack should already have the LiPo connectors. If not, refer to "The On-Board Module")
4) Mount the webcam using the provided mounting braces on the top of the rear rail of SCUTTLE. The webcam should be elevated off of the rail using the twisting extension mount and should also be facing towards the front of the SCUTTLE.
5) Lastly, ensure proper alignment of your front panel and webcam using the diagram provided below in Figure 1.3.

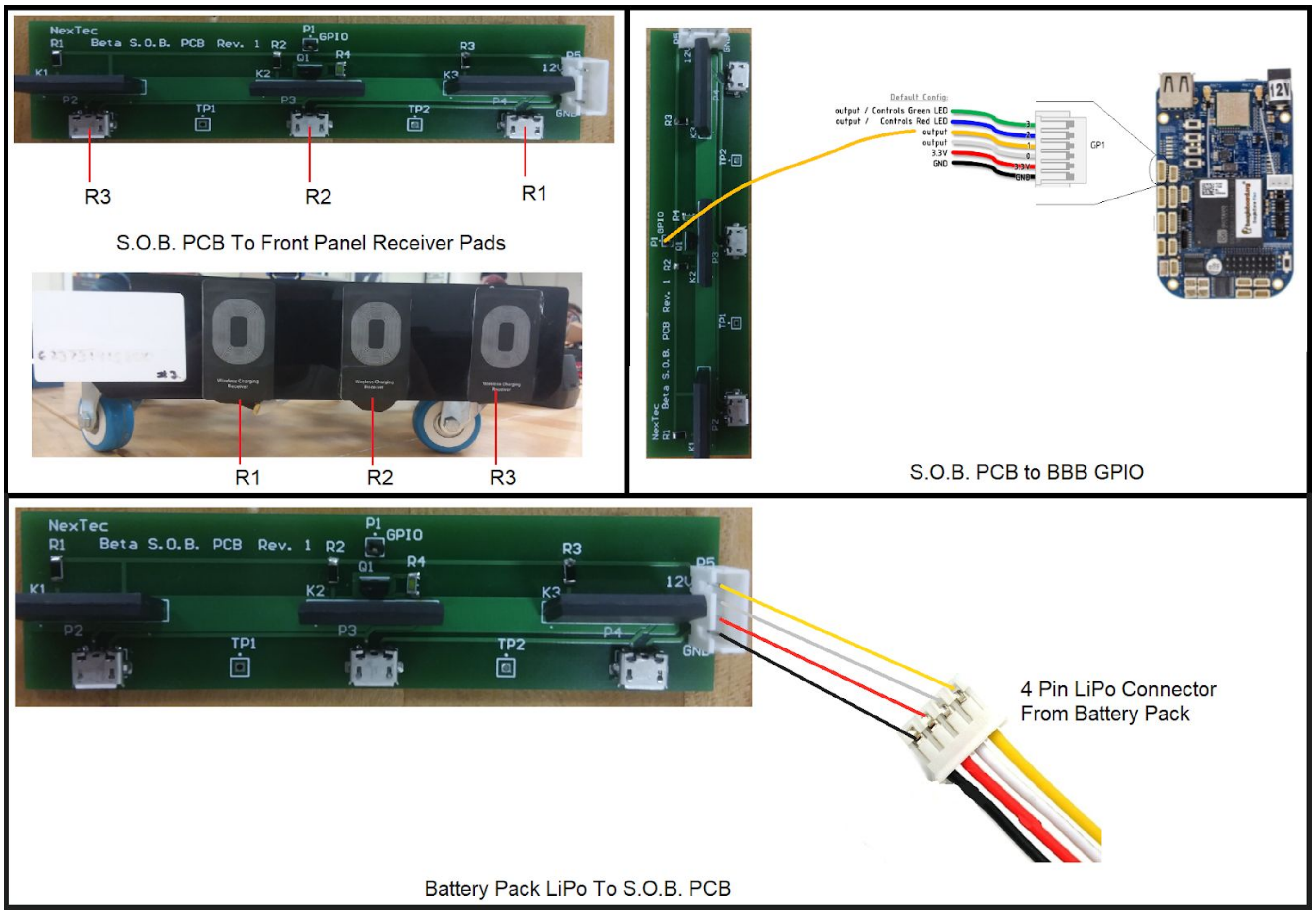

R3
R2
R1
S.O.B. PCB To Front Panel Receiver Pads
R1
R2
R3
Default Config:
output / Controls Green LED
output / Controls Red LED
output
output
3.3V
GND
GP1
S.O.B. PCB to BBB GPIO
4 Pin LiPo Connector From Battery Pack
Battery Pack LiPo To S.O.B. PCB

Figure 1.2: S.O.B. Wiring Diagram

Using Figure 1.3 below for reference, position your SCUTTLE with the attached S.O.B. hardware up against the charging station, ensure that the charging station is NOT powered for now to avoid passing unwanted current to your battery pack. Point 1 in the below figure depicts how the RFID card should be aligned with the RFID reader on the station. It is also important to note that you should position your ID card such that it does not come between the base and receiver pads next to it. It is recommended to use double sided tape to hold your RFID tag so it can be easily moved. Point 2 in the figure depicts how the receiver pads should align with the center of the base pads when up against the station, the etchings on the front panel show where to glue your receiver pads to ensure this alignment. Point 3 shows how the webcam should align with the station when docked, the center of the webcam should be aligned with the center of the orange target ball on the charging station. NOTE: this may not necessarily be the center position relative to the SCUTTLE's rail. After you have completed your alignments, it is safe to power on the charging station.

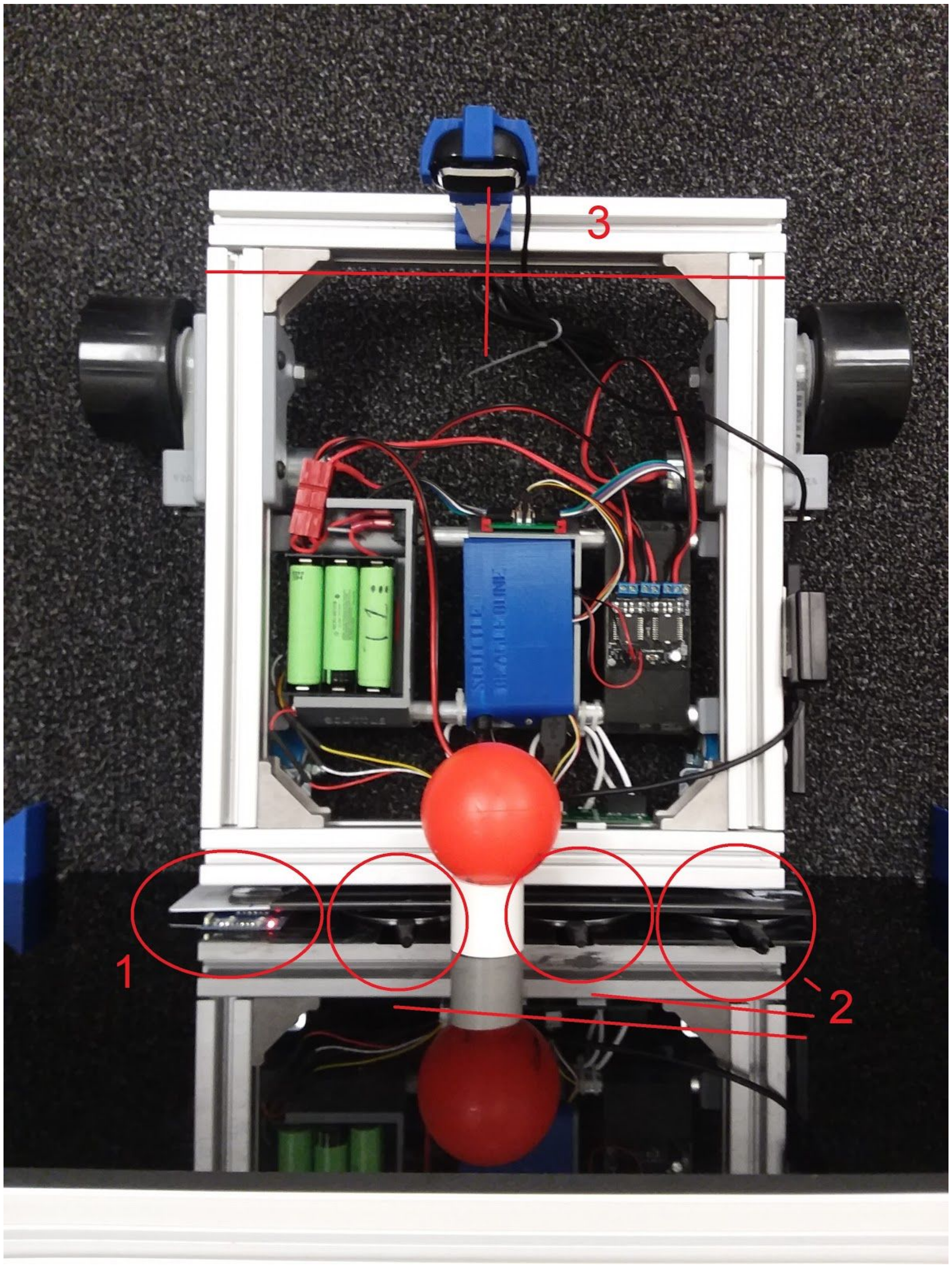


Figure 1.3: S.O.B. and Camera Alignment Diagram

Deploying The Docking Software

Introduction

This section is not intended to be a full overview of the entire software system for the SCUTTLE Power Management System. Rather, this section is intended to provide the steps necessary to simply activate the SCUTTLE's autonomous docking function within your code and to ensure proper calibration of the SCUTTLE's compass and webcam.

Calibrating the Software

Calibrating the webcam only needs to be performed once; it will retain its calibration through power cycles. By default, the camera is calibrated to look for the dark orange ball originally delivered with our prototypes. If the color, material, or lighting properties of the ball should change, the camera will need to be recalibrated for these new target parameters. Calibrating the webcam is already covered in the MXET 300 course and so will not be discussed any further here. The default color target values for the webcam which correspond to the dark orange ball seen in our demonstrations are as follows:

```
color_range = np.array([[0, 76, 255], [41, 224, 255]]) #target for dark orange ball
```

These values are already implemented in the provided software, and therefore do not need to be changed if the desired target is the same dark orange ball.

Properly calibrating the SCUTTLE's onboard compass is a crucial step in ensuring accurate docking. Compass calibration is already covered in the MXET 300 course, however there are two additional steps which must be performed for using the autonomous docking sequence correctly.

First, after properly calibrating the compass using the methods described in the course, the user should acquire the values for the SCUTTLE's target heading when docking at the station. Figure 1.4 below depicts the proper positioning of the SCUTTLE in order to acquire the target heading. It is important that the SCUTTLE be position in the center of its docking spec range and aligned with the receiver pads and ball target. The target heading needs to be acquired in 360 notation and then entered into the docking code. To do this, first open the file "L2_docking_functions.py" in cloud9. Then uncomment the section of code towards the bottom of the file which reads:

```
# while 1:
#     print(GetH360())
#     time.sleep(2)
```

Save the file, then run “L2_docking_functions.py” on the SCUTTLE. Once the SCUTTLE is properly placed according to Figure 1.4, use the printouts in the terminal to determine the target heading in 360 degree notation.

Once the target heading has been determined, enter the value into the file “L3_Dock.py” at line 21, the variable labeled “TH”.

Lastly, keep the “L2_docking_functions.py” code running with the uncommented while loop and turn the SCUTTLE 180 degrees so that now it is identical to the image shown in Figure 1.4 but pointing directly away from the Station. In the same manner as before, use the printouts to find the Station’s heading in 360 degree notation and enter that value into line 22 of the “L3_Dock.py” file at the variable labeled “SH”. In theory, this value would be the opposite of what was found for the target heading which could easily be calculated, however due to inaccuracies with the onboard compass, this is not the case in practice.

With those two values entered, the SCUTTLE is calibrated and ready to run, comment the while loop mentioned earlier and save the “L2_docking_functions” file.

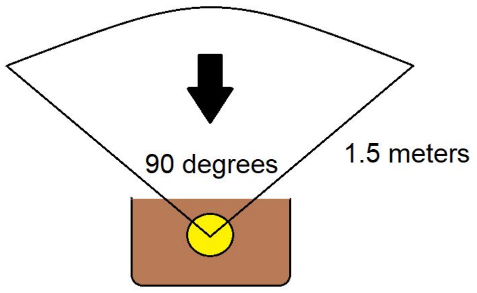


Figure 1.4: Target Heading Calibration (Note: Arrow direction indicates direction of camera facing)

Initiating The Docking Sequence

The docking sequence can be started in a variety of ways, the most common of which however is to call the Dock() function in your script. To do this, ensure your file imports the “L3_Dock.py” file provided. For example:

```
import L3_Dock as Dock
```

With this file imported, running the docking sequence is as simple as calling the function in the example shown below:

```
Dock.Dock() #this tells the SCUTTLE to start autonomously docking.
```

Calling this function will run the entire docking sequence and will also keep the SCUTTLE at the Charging Station until its batteries are fully charged at which point it will then reverse out of the Station and return to the next line of execution at the point where the function call originated.

Using The System Database

The SCUTTLE Power Management System uses the Cayenne MyDevices IoT Database to display Charging Station occupancy status and SCUTTLE battery levels. Using our Cayenne MyDevices account is essential to managing Charging Station and SCUTTLE data.

Connecting to the Cayenne IoT Database

In order to display SCUTTLE battery levels and Charging Station occupancy status, the user must connect to the Nextec Team's online Cayenne dashboard.

1) Go to https://developers.mydevices.com/cayenne/signin/
2) Enter username and password credentials
   Email Address: nextecteam2019@gmail.com
   Password: !Actuallybasic19

Understanding The Dashboards

After logging into the Nextec Team Cayenne MyDevices account, a dashboard with multiple SCUTTLE and Charging Station devices can be seen. Each device corresponds to either a Charging Station or SCUTTLE robot. Each has their own individual dashboard full of different widgets.

1) Charging Station Dashboard

The Charging Station dashboard is responsible for displaying occupancy status information. As seen below in Figure 2.0, the dashboard consists of two different types of widgets. The first widget is the "Vacant" status widget. If this widget is lit up, then the user knows that Charging Station #1 is vacant. The rest of the widgets are the "SCUTTLE #" widgets. These widgets light up when the corresponding SCUTTLE is docked at Station #1.

Extra Charging Station devices and SCUTTLE widgets can be added as needed. This process is described in the "Manipulating the Cayenne IoT Database" section of this user manual.

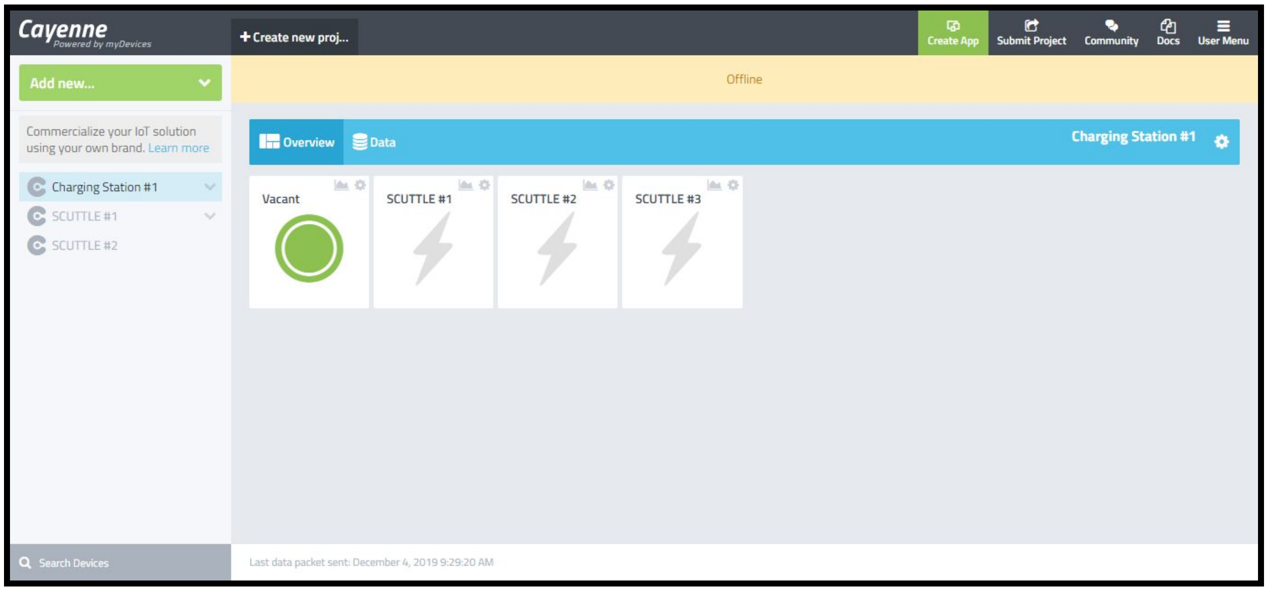


Figure 2.0: Charging Station Dashboard

2) SCUTTLE Dashboard

The SCUTTLE dashboard is responsible for displaying battery level information. As seen below in Figure 2.1, each SCUTTLE's dashboard can be accessed on the left side of the screen. Each dashboard consists of the charging status and battery level widgets. When the charging status widget lights up, this means that the corresponding SCUTTLE is currently being charged. The battery level widget will display the current battery level percentage based on the battery pack's voltage levels.

Extra SCUTTLE devices can be added as needed. This process is described in the "Manipulating the Cayenne IoT Database" section of this user manual.

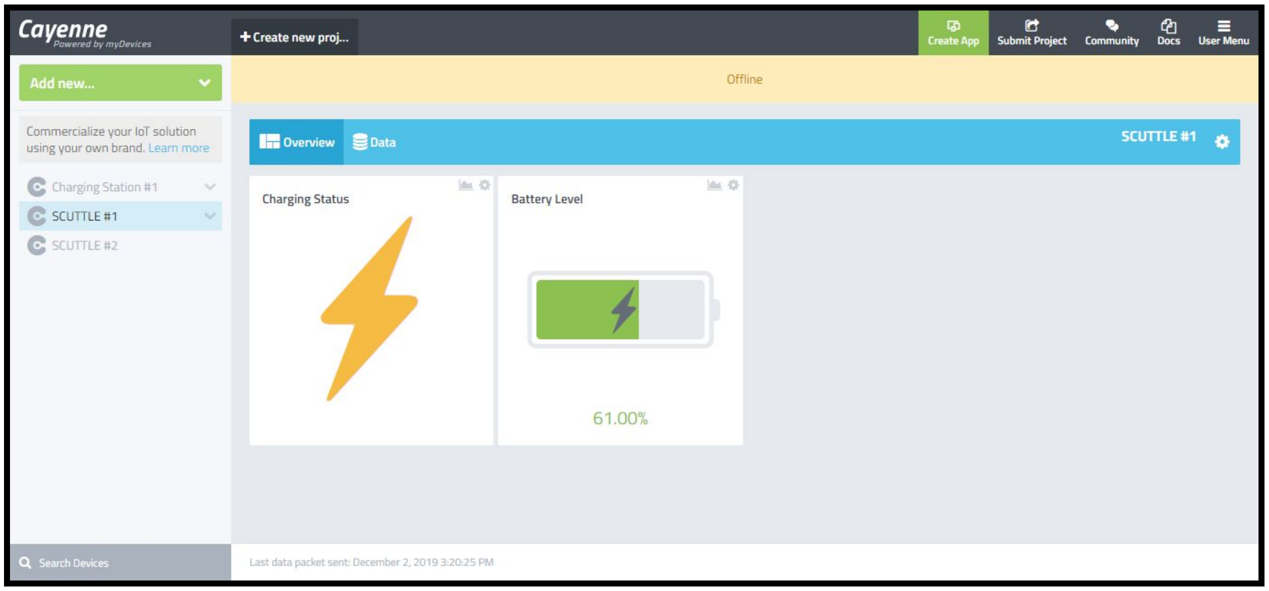


Figure 2.1: SCUTTLE Dashboard

Manipulating the Cayenne IoT Database

This section describes how you can add your own device (Charging Station or SCUTTLE), add new widget displays, and customize widget names/symbols.

1) Adding a new Device

Once logged in, click on the "Add new" button in the top left-hand of the screen, then choose the "Bring Your Own Thing" option.

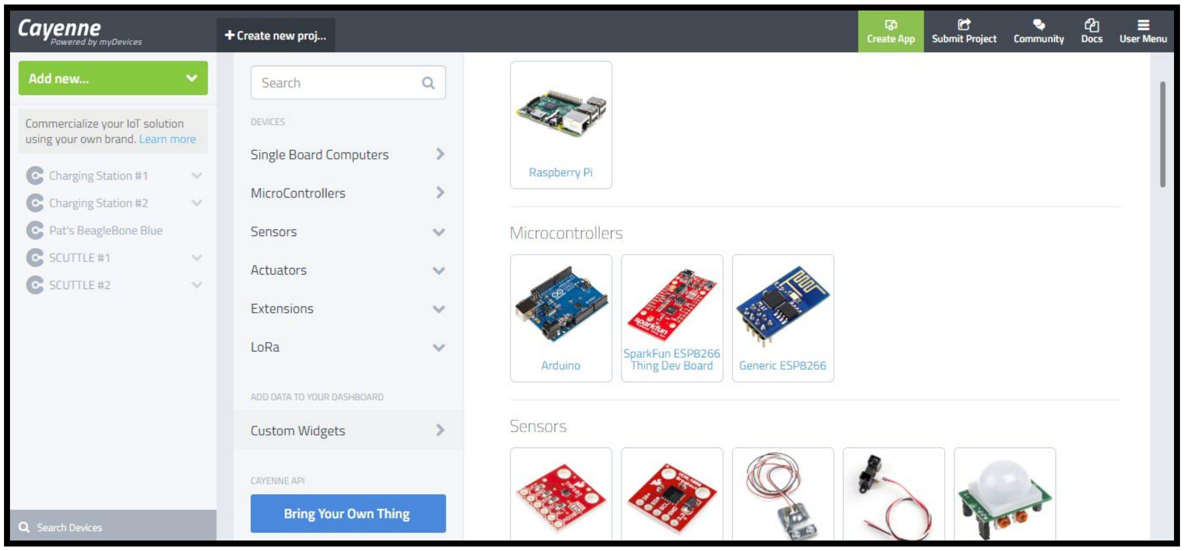
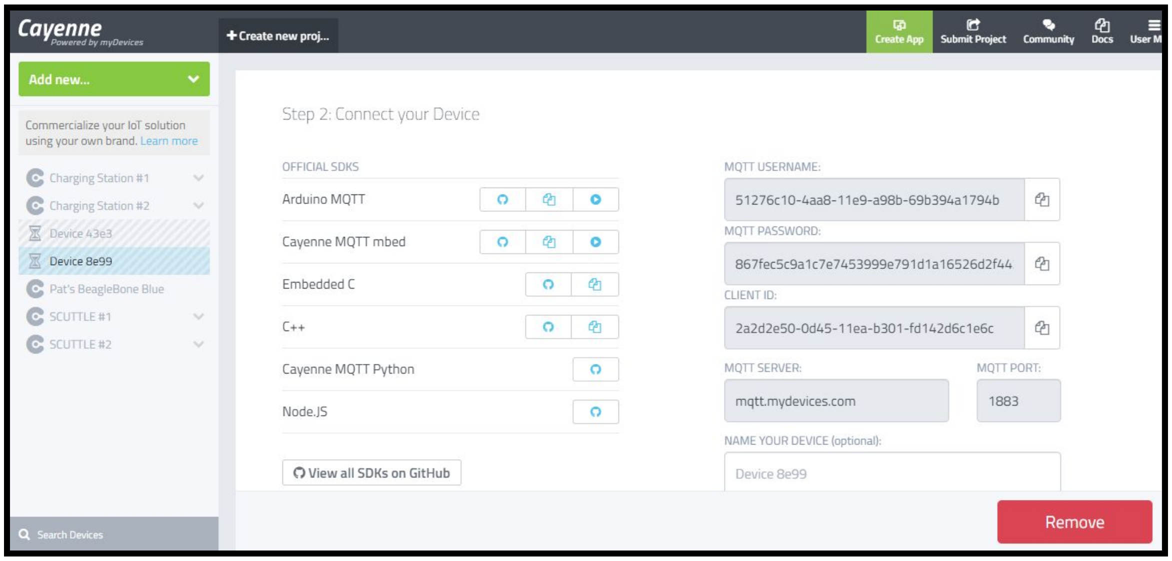


Figure 2.2: Cayenne Add a new Device

Now that you have added a new device, a new device name will show up on the left hand side of your dashboard. This device name should be changed in the bottom right of the screen to either a new Charging Station or SCUTTLE. This screen also provides information on how to connect your device to this new dashboard. The MQTT username, MQTT password, and client ID are used in conjunction with our L1_cayenne code to connect your device. Refer to L1_cayenne for the necessary function calls, and use the MQTT information given to connect to Cayenne.



Commercialize your IoT solution using your own brand. Learn more

Step 2: Connect your Device

MQTT USERNAME: 51276c10-4aa8-11e9-a98b-69b394a1794b

MQTT PASSWORD: 867fec5c9a1c7e7453999e791d1a16526d2f44

CLIENT ID: 2a2d2e50-0d45-11ea-b301-fd142d6c1e6c

MQTT SERVER: mqtt.mydevices.com

MQTT PORT: 1883

NAME YOUR DEVICE (optional): Device 8e99

Figure 2.3: Cayenne MQTT Info

After successfully connecting your BeagleBone Blue to the dashboard using L1_cayenne, your dashboard will change to a blank canvas where new widgets can be added. The MQTT information for your dashboard can always be accessed again by clicking on the cog wheel in the top right hand corner of the dashboard.

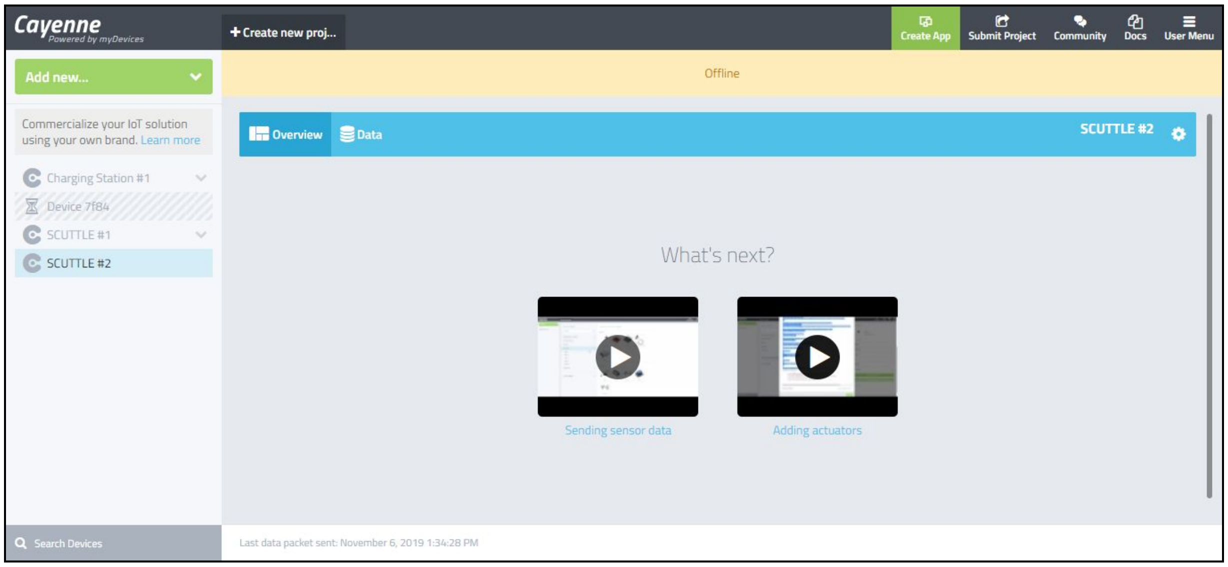


Figure 2.4: Cayenne New Dashboard

2) Adding and Customizing new Widget Displays
   a) Charging Station Widgets

      When adding a new Charging Station to the IoT database, the “Vacant” and “SCUTTLE #” widgets need to be added to complete the charging station dashboard. To do this, begin by running the L3_chargingstation code. Running this code will automatically add the “Vacant” status widget. To make this widget a permanent widget (can later be deleted if needed), click the “+” button in the top right hand corner of the widget. This widget can then be customized by clicking on the settings cog wheel in the top right. From here, the widget name and widget symbol can be changed as necessary. Name the Vacant widget as “Vacant” and choose the green circle as the widget symbol.

      Now that the Vacant widget has been created, you can begin adding as many “SCUTTLE #” widgets as needed. Start by taking your SCUTTLE robot’s RFID tag and scan it using the charging station. The charging station will automatically save this new ID and create a new widget for you. Now that the widget has been added to the dashboard, do as you did before and make the widget permanent by clicking on the “+” button in the top right hand corner of the widget. Same as before, click on the settings button and change the name of the widget to your SCUTTLE’s name. You can then change the symbol to whichever you like. To add more SCUTTLE widgets, simply repeat this process.
   b) SCUTTLE Widgets

      When adding a new SCUTTLE to the IoT database, the “Charging Status” and “Battery Level” widgets need to be added to complete the SCUTTLE dashboard. To do this, begin by running the L3_batterystatus_cayenne code.

Running this code will automatically add these widgets. To make these widgets permanent, click the “+” button in the top right hand corner of each widget. These widgets can then be customized by clicking on the settings cog wheel in the top right. From here, the widget name and widget symbol can be changed as necessary. Name the Charging Status widget as “Charging Status” and choose the lightning bolt as the widget symbol. Similarly, name the Battery Level widget as “Battery Level” but leave the widget symbol alone.

Software Breakdown

Software Overview

The following is a list of all files and functions contained within them which come provided with our project repository.

L1_gpio.py # This code writes and reads from the GPIO pins available on the BeagleBone Blue.

pin_setup(port=None, pin=None, state=None): # A function for setting up pins.

def index_exists(index,i):   # Check that an index exists. Used to check if a pin exists.

check_args(port=None, pin=None, state=None):   # Check that the values passed to our functions are valid

read(port, pin): # Use this function to read an input.

write(port, pin, state): # Use this function to control an output.

L1_cayenne.py # This code is used to setup a connection to Cayenne and publish data to different types of widgets

ConnectionSetup(MQTT_USERNAME, MQTT_PASSWORD, MQTT_CLIENT_ID): #This function Initializes the client and connects to Cayenne

Loop(client): # This function processes Cayenne messages. This should be called regularly to ensure Cayenne messages are sent and received.

digitalWrite(client, channel, value): # This function sends data to Cayenne and creates a digital (boolean 0/1) widget

batteryWrite(client, channel, value): # This function sends data to Cayenne and creates a battery widget

celsiusWrite(client, channel, value): # This function sends data to Cayenne and creates a temperature widget voltageWrite(client, channel, value): # This function sends data to Cayenne and creates a temperature widget

L2_battery.py # This program is used in conjunction with the SCUTTLE On-board Module pack

getBatteryVoltage(): # This function toggles the S.O.B relays and samples barrel jack voltages to determine an accurate battery pack voltage

getBatteryPercentage(BatteryVoltage): # This function is used to convert SCUTTLE's battery voltage to a battery percentage

getChargingStatus(BatteryVoltage): # This function is used to determine if the SCUTTLE is charging by comparing the battery voltage to the voltage when relays are closed

updateBatteryStatusCayenne(client, ChargingStatus, BatteryPercentage): # This function is used to update the battery status to Cayenne

L2_docking_functions.py # This code contains functions for getting the scuttle in front of the station when it is offset by a certain amount

GetSH(): #This function requires a specific NodeRed flow to operate properly. Gist link for NodeRed flow:
https://gist.github.com/TheWhiteCollarPlayers/4efbc1c717bd1d4bc5863db4d5755d41
This function reads the file created by NodeRed which contains the charging station's heading between 0-360

GetTH(SH): # This function takes input of the Station's Heading and returns the target heading. Input and output are in 360 notation.

DriveDP(xdot,tdot,st): #This function takes linear and rotational speed, as well as time, to make the SCUTTLE drive a fixed distance using closed loop driving

GetH360(): #get scaled values from L2 heading, in 360 notation

GetInFront(SH): #This function performs the calculations and drive movements for centering the SCUTTLE directly in front of the station

FindStation(Dir): # This function is used to spin in a circle and search for the station through camera vision

CloseThetaOffset(): # This function is used to align camera vision directly with the station's target

driveStraight(): # This function is for approaching the ball head-on, and stopping when the SCUTTLE is sufficiently close to the target

imageWait(st): #this function is used to mitigate the issue with image processing delays, it takes an input of the amount of time to process images in the buffer stack

CheckForCharging(): # This function is used to check if the SCUTTLE is properly docked and aligned with the charging pads

ChargeAndUndock(client): # This function is used to undock if the batteries are fully charged

L3_batterystatus_cayenne.py # This program is used to continuously update SCUTTLE battery level status to Cayenne

L3_Dock.py #this file contains the function for docking the SCUTTLE at the charging station

Dock(): #this function docks the SCUTTLE at the charging station

L3_gamepaddrive.py #this file allows for basic driving of the SCUTTLE using a gamepad and executing the docking function by pressing the B button

L2_heading_station.py # This program takes magnetometer readings and converts them to be used for the charging station heading.

PubSH(SH): #this function publishes the Station's heading to a topic for the SCUTTLE to receive for docking information, takes in the station heading as an argument (0<SH<360) link for nodered
https://gist.github.com/TheWhiteCollarPlayers/c2b06844936c2a53df7ecf87d130ccbc

getYZ(): # this function returns an average of several magnetometer readings for y and z

scale(axes): # convert raw values to range of [-1 1]

getHeading(myAxes): # convert scaled values to a heading

GetH360(): #scales heading to 360 degree range

L2_RFID_station.py # This program is used for the Charging Station only. These functions are responsible for reading RFID tags, comparing scanned RFID tags to SCUTTLE IDs, and reporting Occupancy status to the Cayenne IoT Database.

scanandcompare(): # This function is used to scan for an RFID tag and compare that tag to the current list of SCUTTLE IDs

checkforunrecognized(id): # This function is used to add a new SCUTTLE ID if the scanned tag is not recognized

updateoccupancystatus(client): # This function is used to update statuses to Cayenne if the statuses have changed

L3_chargingstation.py # This program is used for the Charging Station only. This program is used to continuously update charging station's occupancy status to Cayenne, and update charging station's compass heading to NodeRed topic

Error Codes

ChargingStation Error Codes
901 - L2_heading_station.py, GetH360, #This error code indicates something has gone wrong in the 180 to 360 conversion for heading

SCUTTLE Error Codes
801 - L2_docking_functions.py, GetSH()/GetTH(), #Station heading is greater than expected range

802 - L2_docking_functions.py, GetSH()/GetTH(), #Station heading less than expected range

803 - L2_docking_functions.py, DriveDP(), #The endtime is less than the current time, shouldn't be possible, user likely entered a negative value for stop time

804 - L2_docking_functions.py, GetH360(), #received heading less than expected range, will have improper conversion

805 - L2_docking_functions.py, GetH360(), #received heading greater than expected range

806 - L2_docking_functions.py,GetH360(), #something has gone wrong in the 180 to 360 conversion for SCUTTLE heading, unexpected type most likely or NULL

807 - L2_docking_functions.py, FindStation(), #Target not found within time limit

809 - L2_docking_functions.py, GetSH(), #File read fail

Charging Station Software Setup

In the event that a new charging station is made, this section describes how to make the charging station run L3_chargingstation.py upon bootup.
Note: This “run on bootup” service runs in the background, which means that you can run other python scripts and it will not interfere with the background service. Both scripts will run simultaneously!

1) Creating a background service
   In order to have the charging station code start upon bootup, a background service must be made. Start by navigating to /etc/systemd/system and creating a new service file.
   cd /etc/systemd/system
   sudo nano chargingstation.service
2) Writing your service file
   Next, copy the following script and save it into your new service file. Make sure that the highlighted lines are true for your device. Change the directory (highlighted lines) to wherever your L3_chargingstation.py code is located. Once you have copied the following code into your script, exit and save your script with ctrl+x and y.
   ```
   [Service]
   Type=simple
   ExecStart=/usr/bin/python3
   /home/debian/bin/Nextec/ChargingStation/L3_chargingstation.py
   WorkingDirectory=/home/debian/bin/Nextec/ChargingStation

   [Install]
   WantedBy=multi-user.target
   ```
3) Set up service daemon to run on boot
   Now you can run the following command lines in the terminal to set this service to run in the background on bootup.
   ```
   sudo systemctl daemon-reload
   sudo systemctl enable chargingstation
   sudo systemctl start chargingstation
   ```
4) Checking if it worked

If you followed the steps correctly, the service should be running in the background and will start upon boot up if you reset your device. To test if it is running in the background, simply run the following command line from any directory.

sudo service chargingstation status

If the status is green and active, then you have successfully made the service run in the background. Use ctrl-c to get back to your terminal window.

Note: This “run on bootup” service runs in the background, which means that you can run other python scripts and it will not interfere with the background service. Both scripts will run simultaneously!

Hardware Breakdown

The Charging Station

Electrical / Electronic Assembly

The following is a bill of materials for all electrical / electronic components related to the Charging Station:

Mouser	BeagleBone Blue	95Y0640	1.00
Amazon	SunFounder RFID Kit Mifare RC522 RFID Reader Module with S50 White Card and Key Ring for Arduino Raspberry Pi (Comes with RFID tags for On Board Module)	B07KGBJ9VG	1.00
Amazon	Connector - JST SH 1.0 mm 6-pin pack of 10 (for RFID reader)	B01IZWYK7I	1.00
Amazon	Yi-BEN QI Wireless Charging Pad (includes micro USB cable)	5647450433	3.00
Amazon	5V/6A Power Supply	B0749558XV	1.00
JL PCB	Charging Station PCB (price listed without components)	N/A	1.00
Mouser	USB Connectors WR-COM USB Type A THT Horz Short	614104190121	4.00
Pololu	DC Power Adapter Barrel Jack	1139	1.00

The Gerber files for printing more PCB’s for the Station, are provided with the rest of the documentation for this project. Once these are printed from a boardhouse, we recommend JLCPCB, Figure 3.0 and the information below provide a guide on soldering the components to the board.

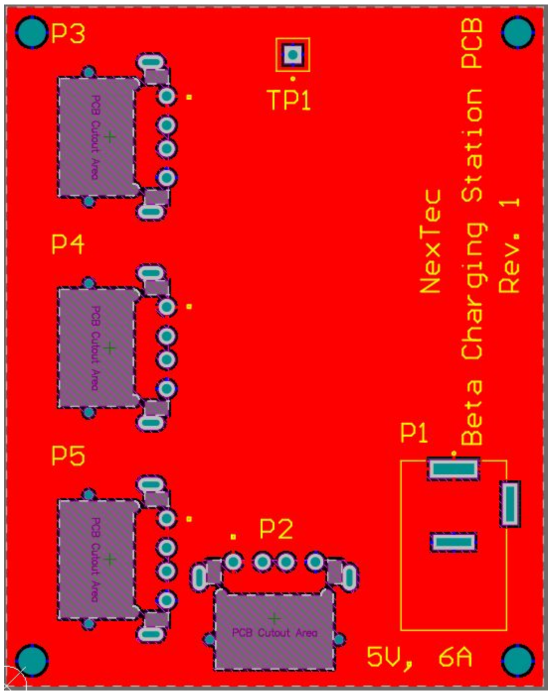


Figure 3.0: Charging Station PCB Layout

Label on PCB - Component For Soldering

P1 - Barrel Jack

P2 - USB 2.0 Female Receptacle

P3 - USB 2.0 Female Receptacle (opening towards board edge)

P4 - USB 2.0 Female Receptacle (opening towards board edge)

P5 - USB 2.0 Female Receptacle (opening towards board edge)

To connect the Station's wireless charging base pads, use a Micro USB to USB cable, much like common phone chargers, and connect the pads to P3-P5. The BeagleBone Blue is also powered via Micro USB to USB by plugging into P2. Figure 3.1 below shows the wiring connection from the Station's BeagleBone Blue to the MFRC522 RFID Interrogator.

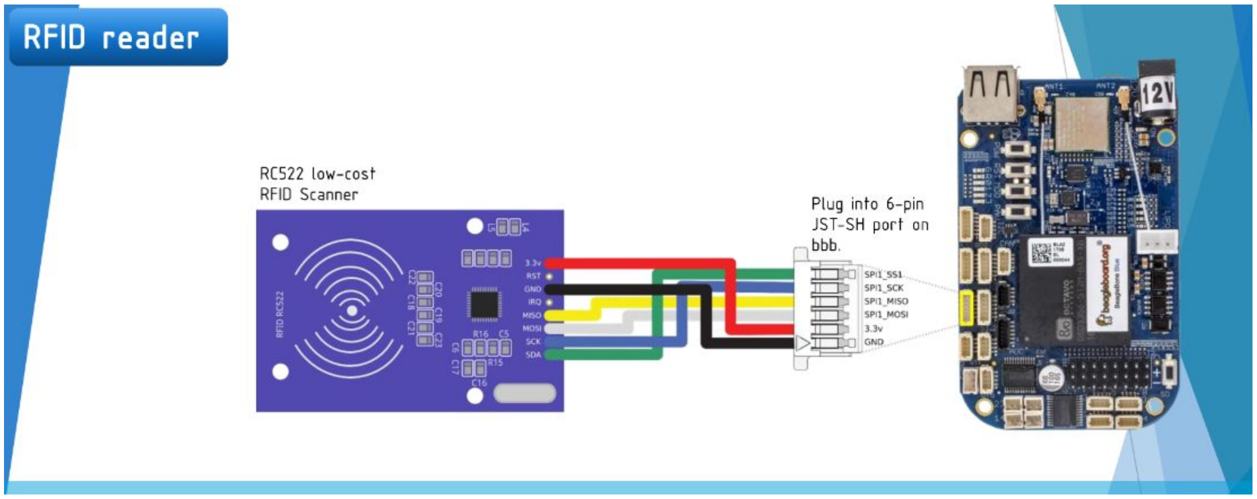


Figure 3.1: BeagleBone Blue to RFID Reader Wiring Diagram

Mechanical Assembly

The following is a bill of materials for all items related to the structural construction and components housing for the Charging Station:

McMaster-Carr	T-Slotted Framing Single Rail, Silver, 30mm (2ft. Length) (for frame)	5537T97	2.00
McMaster-Carr	T-Slotted Framing Single Rail, Silver, 30mm (1ft. Length) (for frame)	5537T98	2.00
Amazon	PZRT 3030 Series Aluminum Profile Connector Set (to connect frame)	B07BQR7WX8	1.00
McMaster-Carr	Steel Pan Head Phillips Screw, M2.5 x 0.45 mm Thread, 6 mm Long, 1 Pack (for RFID reader, Beagle Bone, and Charging Station PCB)	92005A066	10.00
McMaster-Carr	Tapered Heat-Set Inserts for Plastic, Brass, M2.5 x 0.45 mm Thread Size, 3.4 mm Installed Length 1 pack, (for Mounting RFID reader, BeagleBone, and Charging Station PCB)	94180A321	10.00
McMaster-Carr	Screws, M6 x 14 mm (for acrylic) pack of 100	91239A319	8.00
McMaster-Carr	Heat Set Insert, Brass, M5 x 0.80 mm Thread Size, 6.700 mm Installed Length, package of 50 (for target)	94180A361	1.00
McMaster-Carr	M5-0.8 x 40 mm Phillips Pan Head Screw, pack of 10 (for Target)	90116A270	1.00
McMaster-Carr	M5 Zinc-Plated Flat Washers pack of 100 (for Target)	91166A240	2.00
Home Depot	1 in. x 24 in. PVC Sch. 40 Pipe (for mounting Target, cut to 3.5 cm)	202300506	1.00
Amazon	Scotch Indoor Mounting Tape, 0.75-inch x 350-inches, White, Holds up to 10 pounds, 1-Roll (110-LongDC) (for Charging Pads Back Plate)	B009NP1OBC	1.00
Amazon	VELCRO Brand - Sticky Back Hook and Loop Fasteners| Perfect for Home or Office | 5ft x 3/4in Roll | Black (for Charging Pads)	B00006IC2L	1.00
Amazon	Mylec Spring/Summer 6 Pack Balls 6 Pack Spring/Summer Balls, Red/Orange/Pink (Target, Use Orange if using Black Acrylic)	B01LFVL9GG	1.00
Amazon	Charging Station - Front Panel	B01DYSTYBA	1.00
Amazon	Charging Station - Back Panel	B01DYSTYBA	E LEFTOVER ACRYLIC FROM FRONT PAN
Amazon	Charging Station - Charging Pads Back Plate	B01DYSSS12	1.00

The acrylic panels for the front and rear of the Station can be laser cut using the DXF image files provided with the rest of the documentation from this project. Additionally, the “feet” of the Station can be printed using a 3D printer and the STL files provided with the rest of the documentation for this project.
This link shows a walkthrough of the Charging Station assembly: https://youtu.be/B1KRIUH1kig

The On-Board Module

Electrical / Electronic Assembly

The following is a bill of materials for all electrical / electronic components related to the SCUTTLE On-Board module:

Amazon	Wireless Charging Adapter Qi Charger Receiver	B075XC5ZNT	3.00
Amazon	zdyCGTime 4 Inch Cellphone/Tablet USB Micro Male to Female Sync & Charging Extension Cable, 2/pack (connect from On Board Module PCB to Receiver Pads)	B071NR19BQ	3.00
Amazon	LATTECH 15 PCS 4 PIN JST XH Female Connector on One Side, pack of 15 (for modified battery pack)	B07PX14DVW	1.00
Mouser	USB Connectors 5P RECEPTACLE	ZX62D-B-5PA8(30)	3.00
Mouser	Solid State Relays - PCB Mount SinglePole Normally 60V/3.25A DC	CPC1705Y	3.00
Mouser	Thick Film Resistors 0.5watt 1.2Kohms 1% 100ppm	71-RCS12061K20FKE	3.00
Amazon	LATTECH 15 PCS 4 PIN JST XH Female Connector on One Side, pack of 15 (Note: These come in the pack of connetors from Amazon)	B07PX14DVW	-
Mouser	MOSFET N-Chnl 60V	ZVN4206A	1.00
Mouser	Thin Film Resistors - SMD 50kOhm,1206,0.1%,25p pm,125mW,150V	RN732BTTD5002B25	1.00

Figure 3.2 and the information below provide a guide for soldering the components onto the S.O.B. PCB.

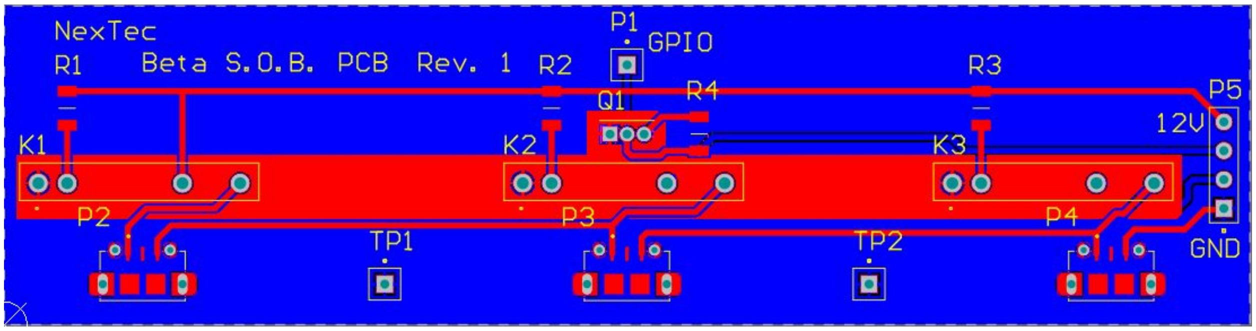


Figure 3.2: S.O.B. PCB Layout

Label on PCB - Component (Good practice to follow this order)

(R1-R3) - 1.2kOhm Resistor

R4 - 50k Ohm Resistor

Q1 - FET (etched circle facing towards P1)

P1 - Single Header Pin

(P2-P4) - Female Micro USB Receptacle (opening facing towards edge of board)

P5 - Female LiPo Connector Receptacle*

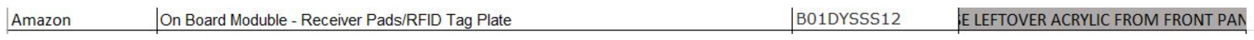
(K1-K3) - Relay

*Note: LiPo receptacle should be soldered such that when plugged in, the cable should have the yellow wire towards the P5 label and the black wire towards the GND label.

This link is a video showing the proper soldering connections to modify the battery pack so that it is compatible with the power management system: https://youtu.be/1plyFnfvYhw

Mechanical Assembly

The following is a bill of materials for all components related to the structural and mounting assembly for the SCUTTLE On-Board Module equipment:



Amazon	On Board Moduble - Receiver Pads/RFID Tag Plate	B01DYSSS12	E LEFTOVER ACRYLIC FROM FRONT PAN

The acrylic panel for the front panel mounting of the receiver pads can be laser cut using the DXF image supplied with the original documentation of our project.

This link is a video walkthrough for assembling the On-Board hardware and mounting it onto the SCUTTLE to ensure its compatibility with the power management system: https://youtu.be/A4b8yb18r9k