Living Architecture Electronics Kit

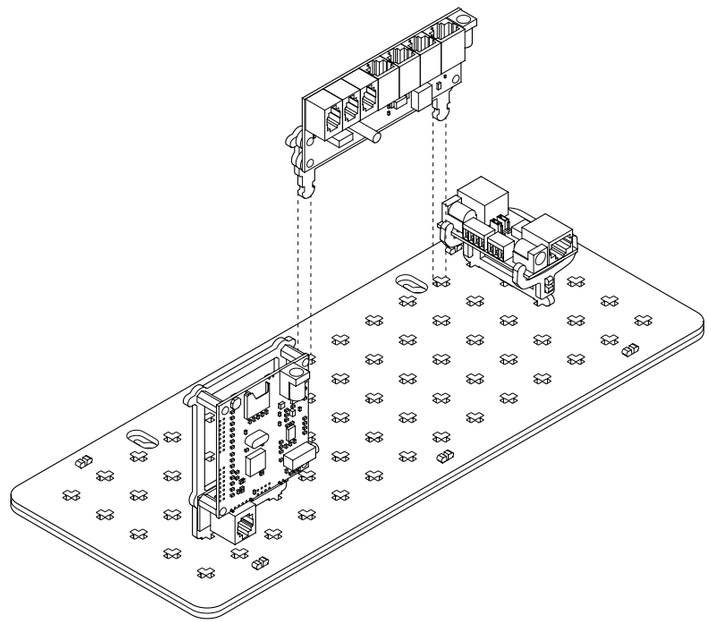
Living Architecture Systems Group

This folio series describes a specialized system of modular distributed electronics control hardware developed by the Living Architecture Systems Group during 2012-2019. The Arduino-based custom electronics provide individual devices containing actuators and sensors with local “intelligence.” Software interfaces are included that support the flexible development of distributed systems that can be easily multiplied and prototyped.

The component designs have been developed as part of a research and creation initiative that is seeking to develop architecture that approaches the qualities of living systems. The scaffold designs that are included within this series support minimal material use and are capable of accommodating multiple components and evolving functions.

Downloadable laser cutting patterns and assembly instructions for electronics and device mounting are included within this publication, supported by open-source Creative Commons licensing that permits adaptation and extension of the electronics control kit.

9 781988 366487 >
Riverside Architectural Press



Living Architecture Electronics Kit

LIVING ARCHITECTURE SYSTEMS GROUP

R LASG

Publisher: Riverside Architectural Press, www.riversidearchitecturalpress.ca

Title: Living Architecture Electronics Kit

Names: Beesley, Philip, 1956-editor. | Living Architecture Systems Group, issuing body.

Description: Series statement: Living Architecture Systems Group folio series | Edited by Philip Beesley. | Includes Index.

Identifiers: ISSN 2562-6787 (Print)
ISSN 2562-6795 (Online)

Production: Philip Beesley, Timothy Boll, Kevan Cress, Lisa Jiang

Editors: Philip Beesley, Timothy Boll

Publication: December 2022
Riverside Architectural Press
11 Dublin Street

Printed in Toronto, Canada.

This book is set in Garamond and Zurich BT.

Social Sciences and Humanities Research Council of Canada
Conseil de recherches en sciences humaines du Canada
Canada

Living Architecture Electronics Kit

Living Architecture Systems Group

Introduction

This folio series describes a specialized system of modular distributed electronics control hardware developed by the Living Architecture Systems Group during 2012-2019. The Arduino-based custom electronics provide individual devices containing actuators and sensors with local "intelligence." Software interfaces are included that support the flexible development of distributed systems that can be easily multiplied and prototyped.

The component designs have been developed as part of a research and creation initiative that is seeking to develop architecture that approaches the qualities of living systems. The scaffold designs that are included within this series support minimal material use and are capable of accommodating multiple components and evolving functions.

Downloadable laser cutting patterns and assembly instructions for electronics and device mounting are included within this publication, supported by open-source Creative Commons licensing that permits adaptation and extension of the electronics control kit.

CONTENTS

SPECIFICATION AND MATERIAL SUMMARY

Type	ID	Name	Specifications and Usage	Manufacturer Part #	Supplier	Default Quantity per Kit*
Power	1.1.1	Wall Adapter	12V 3.0A, 72" Long, Black	WSU120-3000	Digi-Key	1
Raspberry Pi	2.1.1	Rasberry Pi 3 Model B + Custom Assembly	Raspberry Pi 3 Model B - MPU ARM® Cortex®-A53	RASPBERRY PI 3	Digi-Key	1
Node	3.1.1	3.2 Node Controller + Custom Assembly	LAS Node Controller 3.2, 47.6mm x 106.5mm	-	PBAI	1
	3.2.1	Xbee ZigBee Through-Hole	RF TXRX MODULE 802.15.4 WIRE ANT	XB24CZ7WIT-004	Digi-Key	1
Jack Plate	4.1.1	3.2 Jack Plate + Custom Assembly	LAS Jack Plate 3.2, 59.9mm x 31.6mm	-	PBAI	1
	4.2.1	Jack Plate Jumper	CONN JUMPER SHORTING .100" GOLD	QPC02SXGN-RC	Digi-Key	1
	4.3.1	2.0 Cell Breakout Board + Custom Assembly	LAS Cell Breakout Board 2.0, 12.8mm x 76.3mm	-	PBAI	7
Device Module	5.1.1	3.2 High Current Device Module + Custom Assembly	LAS Device Module B 3.2, 34.5mm x 115mm	-	PBAI	1
	5.2.1	3.1 Low Current Device Module + Custom Assembly	LAS Device Module A 3.2, 34.5mm x 91.3mm	-	PBAI	1
Sensors	6.1.1	Infrared (IR) Sensor + Custom Assembly	Sharp IR Distance Sensor, 10-80cm Detection	GP2Y0A21YK	Sparkfun	1
	6.2.1	Sound Detector/Microphone + Custom Assembly	Audio, Electret Microphone Evaluation Board	SEN-12642	Digi-Key	1
Actuator	7.1.1	WAV Trigger Board + Custom Assembly	Audio, Audio Processing Evaluation Board	1568-1115-ND	Digi-Key	1
	7.2.1	Speaker + Custom Assembly	8 Ohm Magnetic Speaker 3W 160Hz ~ 15kHz Top Round, Square Frame 104dB	102-3861-ND	Digi-Key	1
	7.3.1	Adafruit NeoPixel (RGBW) + Custom Assembly	NEOPIXEL RGBW COOL 5050 1=10	2759	Adafruit/Digi-Key	6
	7.4.1	DC Motor + Custom Assembly	STANDARD MOTOR 11605 RPM 5VDC	PPN7PA12C1	Digi-Key	1
	4.3.1	Light-Emitting Diode (LED) + Custom Assembly	LED WHITE 5mm ROUND T/H, connected to Cell Breakout Board	-	PBAI	7
Connections	8.1.1	Barrel Splitter 4-Way	2.1mm ID, 5.5mm OD, Jack to Plug (4)	1528-1455-ND	Digi-Key	1
	8.2.1	Barrel Cable 3'	2.1mm ID, 5.5mm OD, Female to Male, 3'	CA-2216	Digi-Key	1
	8.3.1	Micro USB Cable	USB Splitter to Node, Power to RPi, 0.5m	AK67321-0.5	Digi-Key	1
	8.4.1	NeoPixel Cable	4P4C Flat Line Cable, 12" 26 AWG, NeoPixel Chain, Low Current Device Module to NeoPixel	-	PBAI	6
	8.4.1	Cell Breakout Board Cabling	4P4C Flat Line Cable, 12" 26 AWG, High Current Device Module to NeoPixel	-	PBAI	6
	8.5.1	IR Sensor Cable	6P6C Flat Line Cable, 12", Device Module to IR	-	PBAI	1
	8.5.1	WAV Trigger Cable	6P6C Flat Line Cable, 12" 26 AWG, Device Module to WAV	-	PBAI	1
	8.5.1	MIC Sensor Cable	6P6C Flat Line Cable, 12" 26 AWG, Device Module to MIC	-	PBAI	1
	8.6.1	RJ45 Cabling	8P8C Flat Line Cable, 12" 26 AWG, Device Module to Node Controller, Jack Plate Chain	-	PBAI	3
	8.7.1	USB Splitter	USB Splitter Cable, One Male - Two Female Ports	RR-USB2-SPLITTER	USB Firewire	1
	8.4.1	4P4C Plug	Cell Breakout Cable End	A-MO 4/4 F50	Digi-Key	per # of cables
	8.5.1	6P6C Plug	Sensor, WAV Trigger Cable End	A-MO 6/6 F50	Digi-Key	per # of cables
	8.6.1	8P8C Plug	RJ45 Cable Ends	A-MO 8/8 F50	Digi-Key	per # of cables
Setup	9.1.1	Xbee Explorer USB Dongle	USB to Raspberry Pi for setup of Digi Xbee	1568-1116-ND	Digi-Key	1
	9.2.1	USB Memory Card with Code Package	PBAI Company Information, Code Package	-	PBAI	1
	9.3.1	MicroSD Card for Raspberry Pi	16GB SD Card NOOBS Software	1690-1002-ND	Digi-Key	1
	9.4.1	MicroSD Card to USB Reader	MicroSD card slot to USB to computer for WAV tracks	1568-1414-ND	Digi-Key	1
	9.5.1	MicroSD Card for WAV Trigger	MEM CARD MICROSDHC 8GB CLS10 MLC, WAV Trigger Track Storage	P17028-ND	Digi-Key	1
	9.6.1	Acrylic Tray	Mounting Tray for Electronic Components	-	PBAI	1
	9.7.1	User Manual	Setup Instructions and Kit Information	-	PBAI	1
	9.8.1	Bag of Acrylic Locking Clips	Locking Clips for Electronic Components	-	PBAI	1
	9.9.1	Packaging	Cardboard Box and Foam Insulation	-	PBAI	1

SAFE HANDLING AND CARE OF ELECTRONICS

When handling any electronic devices or printed circuit boards (PCBs), there are several rules to follow to ensure your boards are in an excellent working condition.

1. Keep your electronic devices on the acrylic mounts attached to the boards. This serves as a protective barrier when handling the electronics near other metal objects. If your electronic device touches an electronic surface, it could short-circuit connections within your board, rendering the device to be useless.

2. Do not bring devices close to water. Ensure the devices are operating in a low-humidity, dust-free environment.

3. It would be good practice to put the electronics back into the box, sealed by the foam, when not in use. This ensures there is no static discharge on the electronic components, when handling the electronics.

4. Ensure switch on WAV triggers are switched to "RUN". If the WAV triggers are set to "LOAD", the necessary functionalities of the WAV Trigger to run in your system will not work.

5. **IMPORTANT RULE FOR TEENSY DEVICES:**
   Based on previous experiences using the microcontroller, it was found that the Teensy is capable of short-circuiting in an altogether different circumstance, apart from the ones labelled above. Please ensure that all the cabled/mod-jack connections within your system are plugged in BEFORE powering on the Node Controller and Device Modules in your system. If you wish to change the orientation, or unplug any peripheral devices from the Device Modules or Node Controllers, please POWER OFF the system before unplugging the mod-jack connections. If you plug and unplug peripheral devices, while the Node Controller or Device Modules are powered by USB or external barrel jack connections, there is a risk of shorting out the Node Controller. there is no static discharge on the electronic components, when handling the electronics.

6. The wall adapter plug allows for an input voltage between 100 V – 240 V. Therefore, a plug converter would be required for use outside of North America. A voltage converter is not required for this setup. The maximum current drawn by the wall adapter is 1.3 A. In terms of the output, 12 V can be applied to the system. This applied voltage falls within the range of operating voltages for all device modules (see 7.).

7. Each device module has a power restriction, set by the voltage regulators. See the PCB Designs of Core Components section for part numbers of converters. Offering a high-level summary of the power restrictions for each module:
   a. Node Controller: Can take up to maximum of 25 V input applied voltage. Minimum input applied voltage for function is 5 V. Output applied voltage to all ports is adjusted to 5 V. Maximum output current, when all ports connected, is 1.5 A drawn.
   b. High Current Device Module: Can take up to maximum of 14 V input applied voltage. Minimum input applied voltage for function is 4.5 V. Output applied voltage to all ports (in parallel) is adjusted to 5 V. Maximum output current, when all ports connected to an active load, is 10 A. Any current draw in excess of 10 A from a load will not allow for actuation / operation of all active connected loads.
   c. Low Current Device Module: Can take up to maximum of 25 V input applied voltage. Minimum input applied voltage for function is 5 V. Output applied voltage to all ports (in parallel) is adjusted to 5 V. Exception: Port C applies 3.3 V on output. Maximum output current, when all ports connected to an active load, is 3 A. Any curent draw in excess of 3 A from a load will not allow for actuation / operation of all active connected loads
   d. Jack Plate: Can take up to maximum of 25 V input applied voltage. Minimum input applied voltage for function is 5 V. Output applied voltage to two screw terminals (in parallel) is adjusted to 5 V.
   Maximum output current when actuator and sensor connected to screw terminals, is 3 A. Any current draw in excess of 3 A from an actuator will not allow for actuation / operation of the actuator.

8. Note for Low Current Device Module: Ensure switch on board is set to "NP" (for correct applied voltage (5V) to Port D

DEVICE DESCRIPTIONS

POWER

1.1.1 Wall Adapter – Main AC/DC Power Supply with male barrel jack connection (12V 3A 36W).

BOARDS/CORE COMPONENTS

2.1.1 Raspberry Pi 3 model B – A single-board computer used to program high-level behaviour of connected Node Controllers used for calculations and processing in many installations and can be used to control nodes to vary behaviours and systems

Main Features:
-on-board Linux operating system

Connectivity:
-USB for connection to multiple Node Controllers
-ethernet port for connection to internet

Power:
-5V
-micro USB cable (recommended minimum 2A Wall Adapter, computer USB port will suffice if using this method)

3.1.1 3.2 Node Controller – Enables batch programming and serial communication between Raspberry Pi and Teensy Microcontroller.

Main Features:
-Teensy 3.2 Microcontroller
-integrated XBee ZigBee module
-pinout ports through RJ45 Terminals through USB

Connectivity:
-four data ports can be connected to peripheral boards using an RJ45 Cable

Power:
-7.5V-12V
-barrel jack or microUSB port located on Teensy 3.2 (ensure common ground for connected peripheral)
-on-board switch to switch between 5 V USB power (e.g. from laptop / Raspberry Pi) or external power provided by 12 V wall adapter plug. Can be used as ON/OFF switch if only one of the power sources are connected.

3.2.1 XBee ZigBee Wireless Module - Offers wireless communication between Nodes and Raspberry Pi; soldered to 3.2 Node Controller board

4.1.1 3.2 Jack Plate – A peripheral board that can be daisy-chained to control a single actuator and read a single sensor, using screw terminal connections.

Main Features:
-daisy chainable up to 4 Jack Plates
-screw terminal for sensor and high current actuator
-uses MOSFET to power up to 3A actuator

Connectivity:
-connected to Node Controller using RJ45 Cable with J IN (DATA IN) Port

Power:
-7.5V-12V
-barrel jack

The four screw terminal block on the Jack Plate is used for high current actuators and LEDs. As indicated on the Jack Plate (see PCB Designs in later section), a two-lead wire LED (not provided with this kit) can be connected to the terminals labelled "LED -" and "+". Ensure positive end of LED is connected to "+" terminal, for function. A high current actuator, such as a shape memory alloy (not provided with this kit) can be connected to the terminals labelled "A-" and "+". An LED and an actuator can both be connected to the "+" terminal (e.g. LED indicates current draw by high current actuator). The actuator must also have only two lead wires for operation.

The three screw terminal block on the Jack Plate is used for sensors. It will not be necessary for any of the research experiments, since specifically the IR sensor can be connected, modularly, to a Low Current or High Current Device Module. The PCB Designs section offers a more diagrammatical explanation for the power distribution to the screw terminals of the Jack Plate.

Jumpers on Jack Plate should be kept on "DIG" (DIGITAL) connection if controlling actuators connected to screw terminals "+" and "A-" on four screw terminal block. Do not remove jumpers from these set positions if programming for actuation. The set positions are the second and third pins of the male header pins (directly above and below the "DIG" label, respective to the jumper block).

4.2.1 2.0 Cell Breakout Board – A peripheral board that controls external actuators using screw terminals, and has an on-board LED.

5.1.1 3.2 Device Module A Low Current – A peripheral board that utilizes I/O pins from the Node Controller to enable serial and I2C communication to:

1. control actuators (eg. speaker and NeoPixels)
2. read sensors (eg. infrared distance sensors and microphones)

Main Features:
-WAV Trigger control
-microphone and IR
-distance sensor interface
-NeoPixel control

Connectivity:
-I2C and serial pinouts
-uses I/O Pins of Teensy to control various peripherals through digital, analog and communication signals
-connected to Node Controller using RJ45 Cable with DATA Port
-Port D works like the other 4P connections on board, and also includes a switch allowing you to chnage the applied voltage to a peripheral device (see schematic).

Power:
-7.5V-12V
-barrel jack

5.2.1 3.3 Device Module B High Current – A peripheral board that utlizes I/O pins from the Node Controller to control high current acutators such as shape memory alloys, lights and read sensor values.

Main Features:
-high current actuator control
-analag input sensor interface
-uses a MOSFET to regulate up to 5A per actuator channel, maximum of 8A total across all channels

Connectivity:
-connected to Node Controller using RJ45 Cable with DATA Port

Power:
-7.5V-12V
-barrel jack

SENSORS

6.1.1 Microphone Sensor – A sensor that detects loudness and can return audio signals to the Node Controller.

Main Features:
-analog signal providing evelope and raw audio signal
-10-bit resolution

Connectivity:
-connected to Device Module A Low Current using Mic Sensor cable

Power:
-powered by Device Module A Low Current, 3.3V, using Mic Sensor cable

6.2.1 Infrared (IR) Distance Sensor – A sensor that detects how far an object is in front of it.

Main Features:
-analog signal providing 10-80cm range, 10-bit resolution for Teensy

Connectivity:
-to Device Module A Low Current & Device Module B High Current using 3P IR Sensor cable
-to Jack Plate by removing 4P Phoenix and using screw terminals with three of four pins

Power:
-powered by peripheral board, 5V using IR Sensor cable

ACTUATORS

7.1.1 WAV Trigger Board – An expansion board that houses circuitry to amplify and play recorded tracks to a speaker.

Main Features:
-on-board amplifier for external speaker
-polyphonic- supports up to 14 independent, simultaneous stereo tracks
-micro SD card memory
-serial control
-Visit *https://robertsonics.com/wav-trigger/* for in-depth information and specifications about WAV Trigger embedded audio player

Connectivity:
-connected to Device Module A Low Current using WAV Trigger cable with 4P A or 4P B Port

Power:
-powered by Device Module A Low Current, 5V through WAV Trigger cable

7.2.1 DC Motor - An actuator that converts direct current electrical energy to rotary mechanical energy. It is attached to a cell breakout board.

7.3.1 Speaker – An actuator connected to and powered by the WAV Trigger board.

7.4.1 NeoPixel – An actuator that produces a RGB light with a cool white, natural white or warm white default setting.

Main Features:
-addressable
-one wire control
-chainable

Connectivity:
-connected to Device Module A Low Current using 4P NeoPixel cable

Power:
-powered by Device Module A Low Current using 4P NeoPixel cable

CONNECTIONS

8.1.1 4-Way Barrel Splitter Cable – A connector cable used to distribute power to boards from the wall adapter

8.2.1 3' Barrel Extension Cable – A connector cable used to extend power distribution.

8.3.1 Micro USB Cable – A connector cable used to connect the Raspberry Pi to Node Controllers or to power the Raspberry Pi. Can be used to power and send code to Teensy microcontrollers on Node Controllers.

8.4.1 RJ9 Cable - A 4P4C data cable used to connect to actuators such as NeoPixels and Cell Breakout Boards. The 4P4C cable connects to ports C, D, E, F, G and H of the High Current Device Module. It also connects to ports D, E and F of the Low Current Device Module. These cables can only power peripheral devices.

8.5.1 RJ11 Cable - A 6P6C data cable used to connect to sensors such as IR Sensors, Microphone Sensors and the WAV Trigger Board. The 6P6C cable connects to ports A and B of the High Current Device Module. It also connects to ports A, B and C of the High Current Device Module. These cables can both power peripheral devices, while sending and receiving data.

8.6.1 RJ45 Cable - A 8P8C data cable used to connect the Node Controller to the Low Current, High Current and 3.2 Jack Place device modules. The 8P8C cable connects to ports P0, P1, P2, and PS of the Node Controller. They also connect to the DATA ports of the Jack Plate and High Current and Low Current Device Modules. These cables are mainly used for transmitting data (both sending and receiving) directly between the device module and Teensy microcontroller.

* The number of wires within each cable (4P, 6P or 8P) can be determined by counting the number of brass pins on the modular jack connectors, which are crimped to the cables. The width of the cables increases from the 4P4C cables to the 8P8C wires.

SETUP

9.1.1 XBee USB Dongle - A board that connects to Raspberry Pi to set up the XBee for wireless communication between the Node Controller and Raspberry Pi. Included as part of accessories and cables of the Desktop Kit.

SOFTWARE

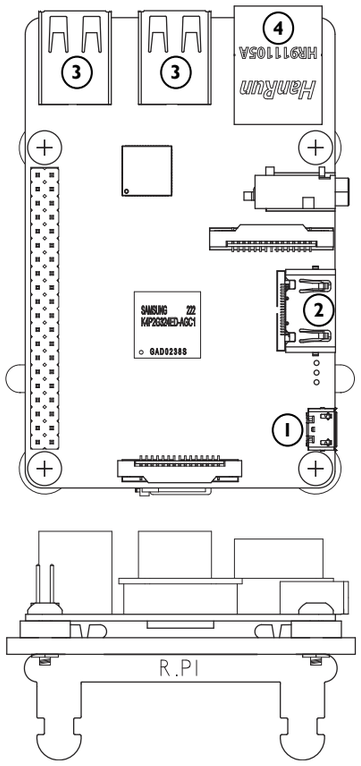
GitHub Repository - https://github.com/pbarch/1712-Series-3.2-Desktop-Electronics-Kit

Note: The basic code package can be found on the USB stick that is included in this kit.

DEVICE DIAGRAMS

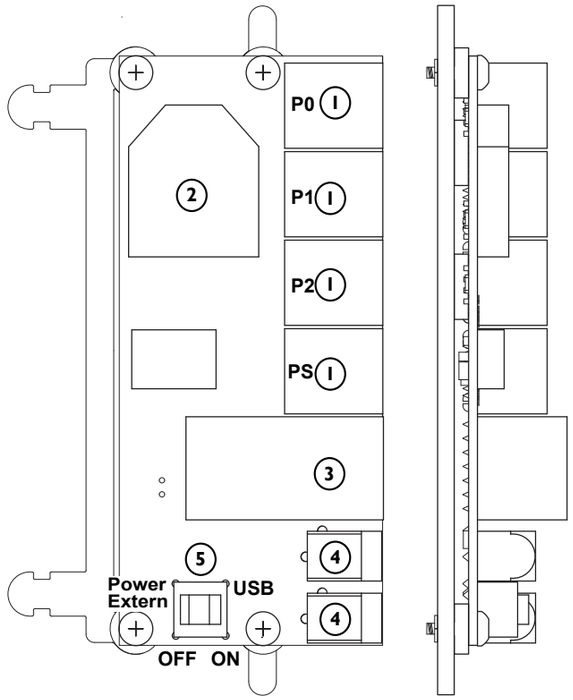
Boards/Core Components

2.1.1 Raspberry Pi 3 Model B



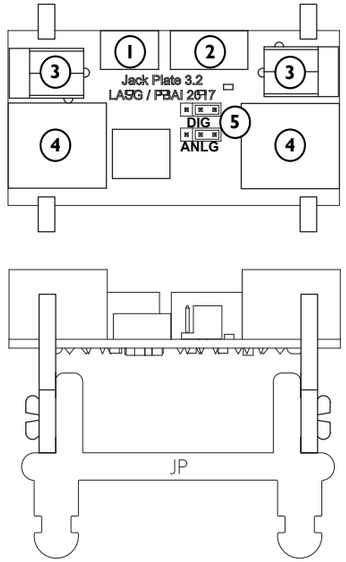
1. Micro USB connection
2. HDMI connection
3. USB ports
4. Ethernet internet connection

3.1.1 3.2 Node Controller



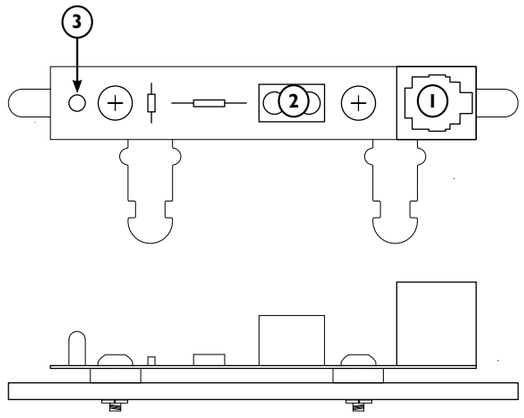
1. 8P8C modular jack connection
   -Ports P0, P1, P2, PS
2. XBee ZigBee module
3. Teensy 3.2 Microcontroller
4. Power barrel jack connection
5. USB/External power switch

4.1.1 3.2 Jack Plate



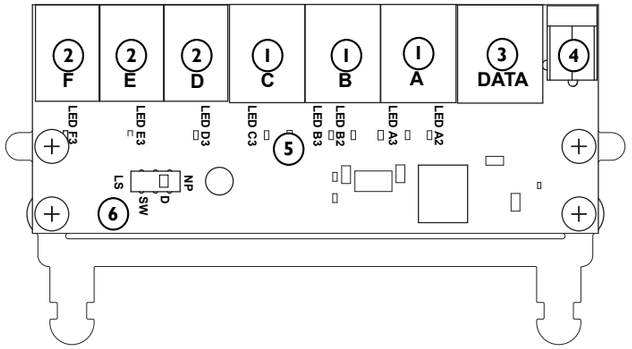
1. 3 screw terminal connection
2. 4 screw terminal connection
3. Power barrel jack connection
4. 8P8C modular jack connection
5. Jumper connections

4.2.1 2.0 Cell Breakout Board

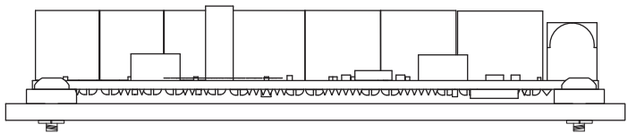


1. 4P4C modular jack connection
2. 2 screw terminal connection
3. LED

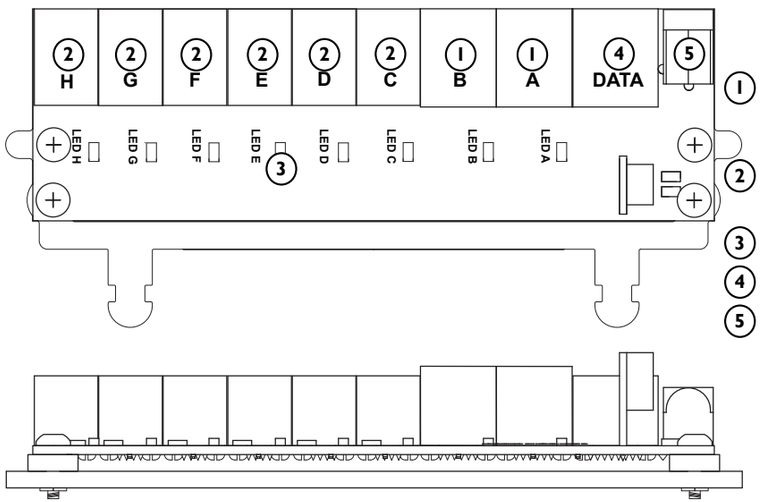
5.1.1 3.2 A Low Current Device Module



1. 6P6C modular jack connection
   -Ports A,B,C
2. 4P4C modular jack connection
   -Ports D,E,F
3. 8P8C modular jack connection
4. Power barrel jack connection
5. LED Signal
6. Port D Voltage Switch



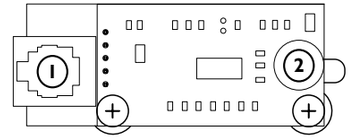
5.2.1 3.3 B High Current Device Module

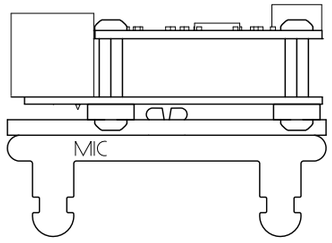


1. 6P6C modular jack connection
   -Ports A,B
2. 4P4C modular jack connection
   -Ports C,D,E,F,G,H
3. LED signal
4. 8P8C modular jack connection
5. Power barrel jack connection

Sensors

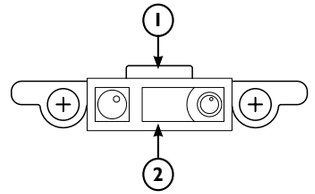
6.1.1 Microphone Sensor

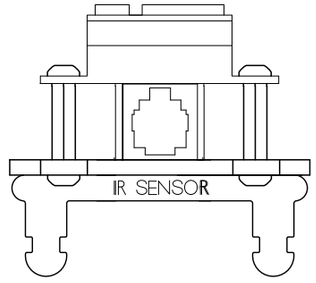




1. 4P4C Modular Jack
2. Sound detector

6.2.1 IR Sensor Modular Board

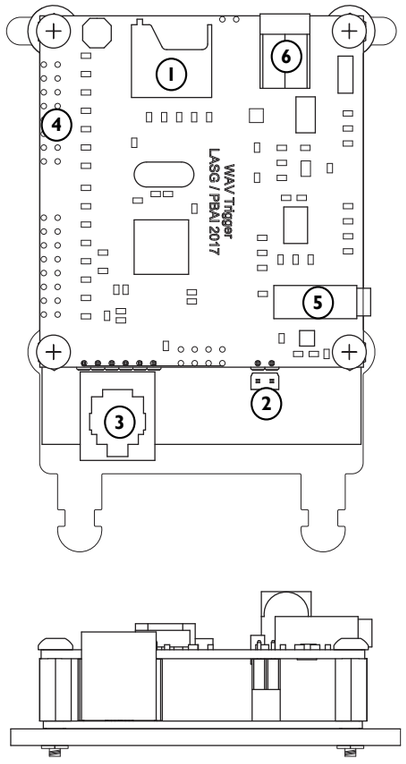




1. 4P4C Modular Jack
2. Distance Sensor

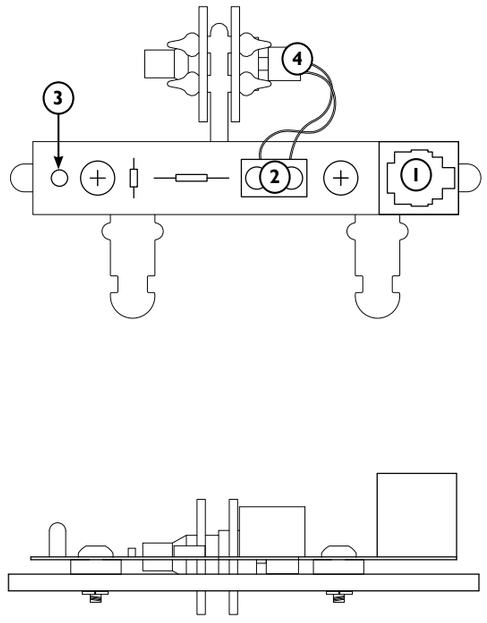
Actuators

7.1.1 WAV Trigger Board



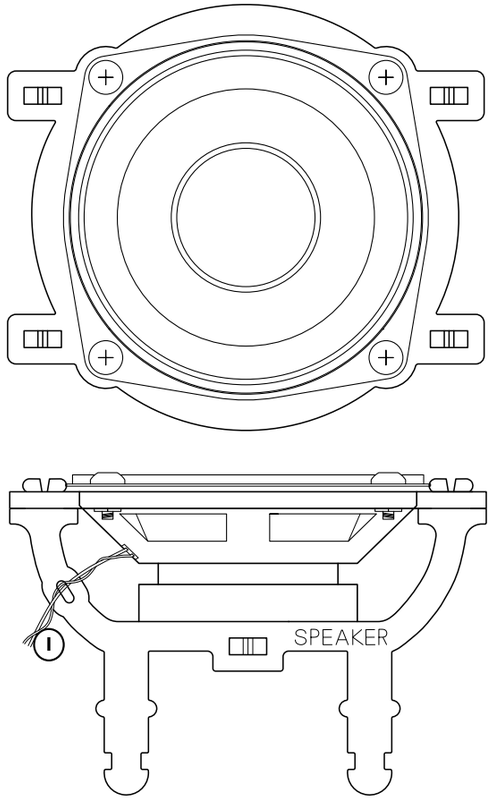
1. SD card slot
2. 2P phoenix connection
3. 6P6C modular jack connection
4. Stereo track switching
5. 3.5mm audio jack
6. External power jack

7.2.1 2.0 Cell Breakout Board with DC



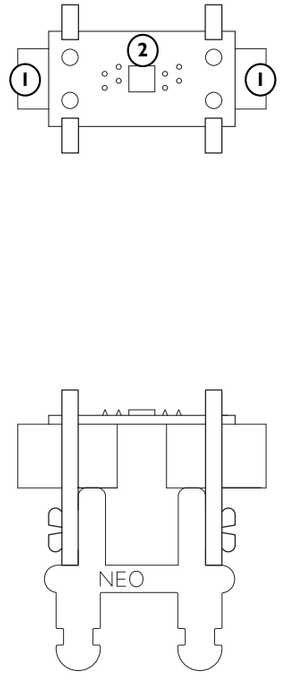
1. 4P4C modular jack connection-
2. 2 screw terminal connection
3. LED
4. DC Motor

7.3.1 Speaker



1. 2P Phoenix connector

7.4.1 NeoPixel Board

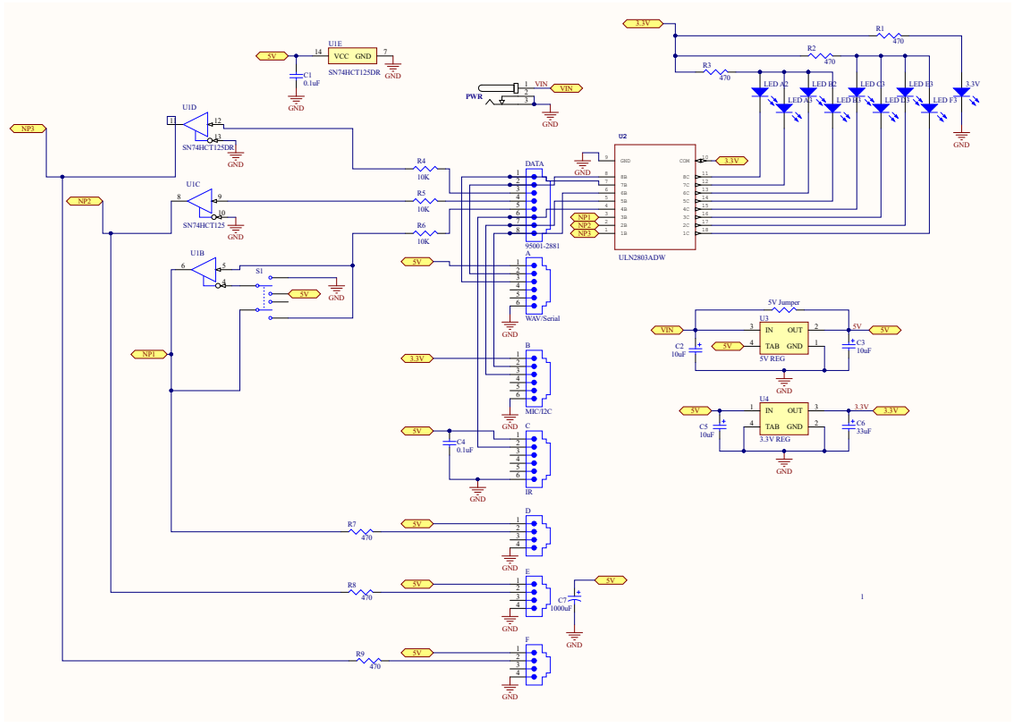


1. 4P4C modular jack
2. NeoPixel

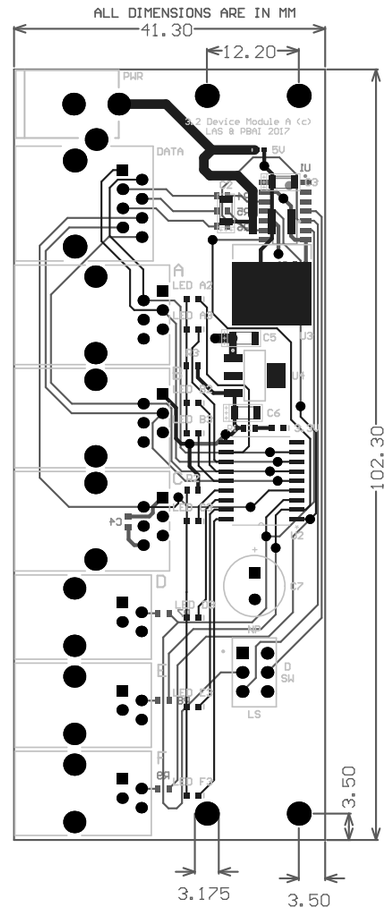
PCB DESIGNS OF CORE COMPONENTS

3.2 Device Module A Low Current

PCB Schematic with I/O Pin Diagram

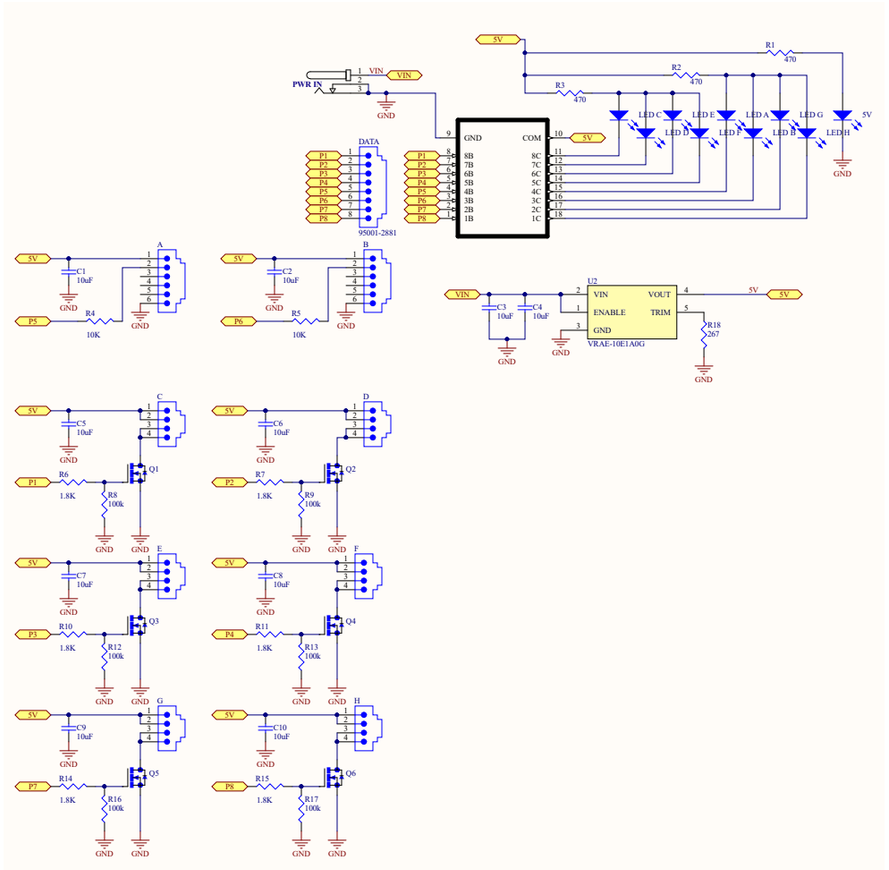


PCB Diagram with Trace Paths

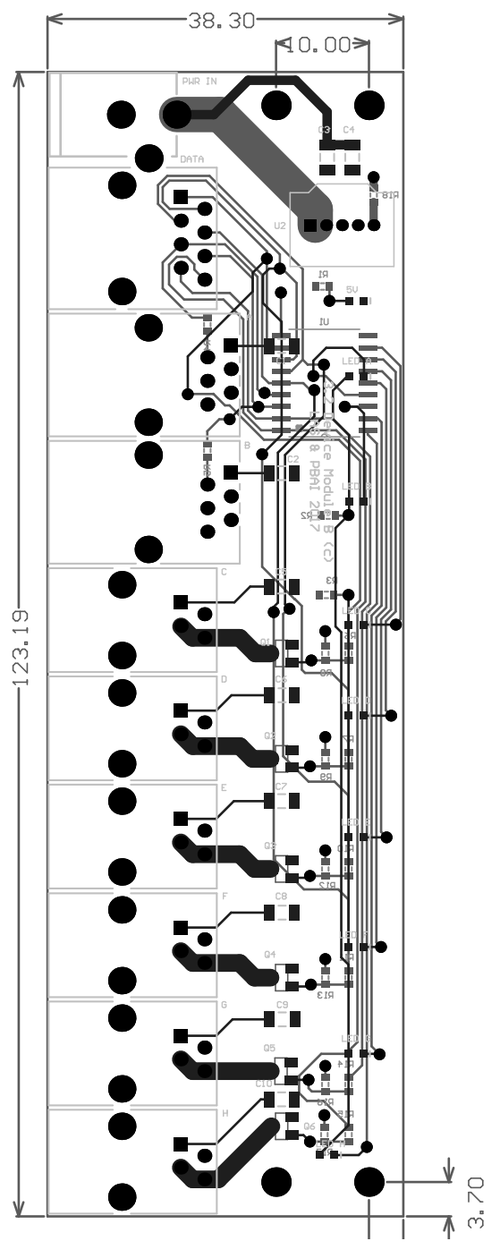


3.3 Device Module B High Current

PCB Schematic with I/O Pin Diagram

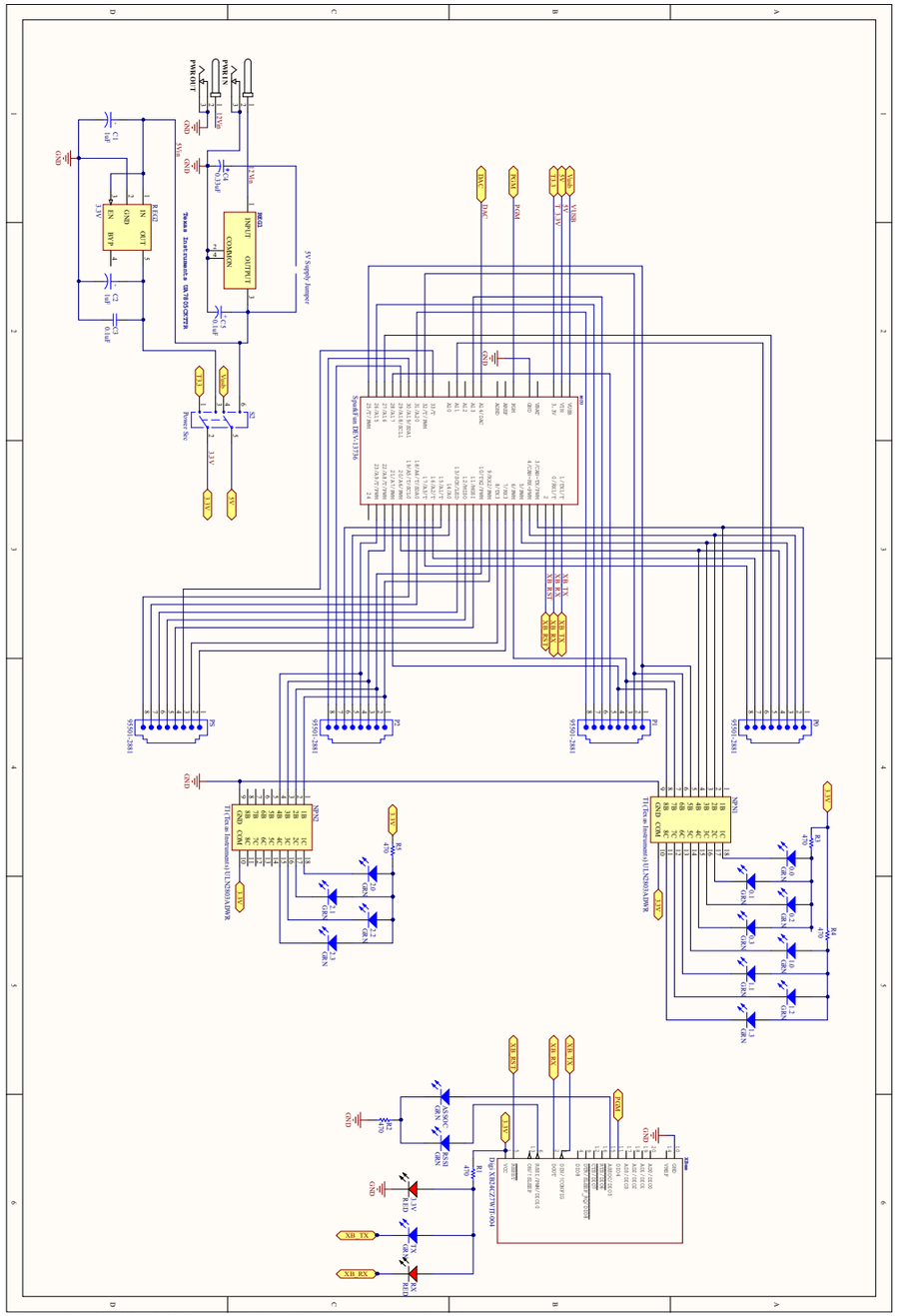


PCB Diagram with Trace Paths

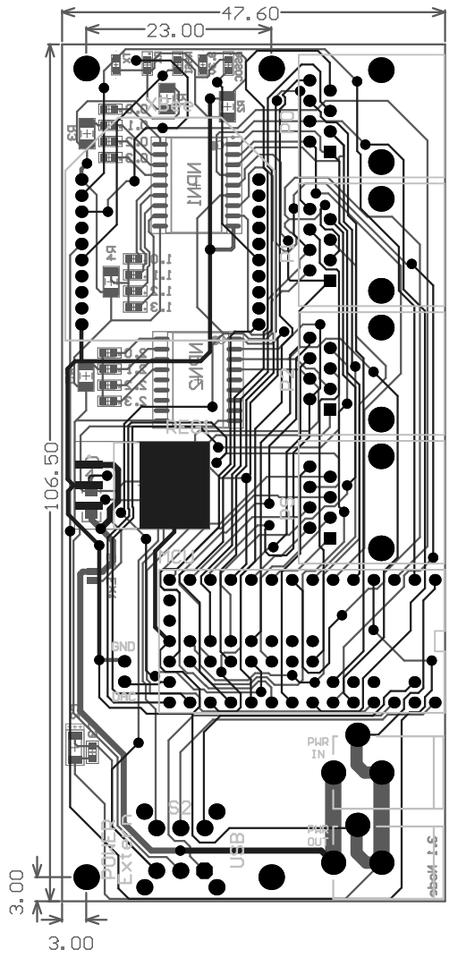


3.2 Node Controller

PCB Schematic with I/O Pin Diagram

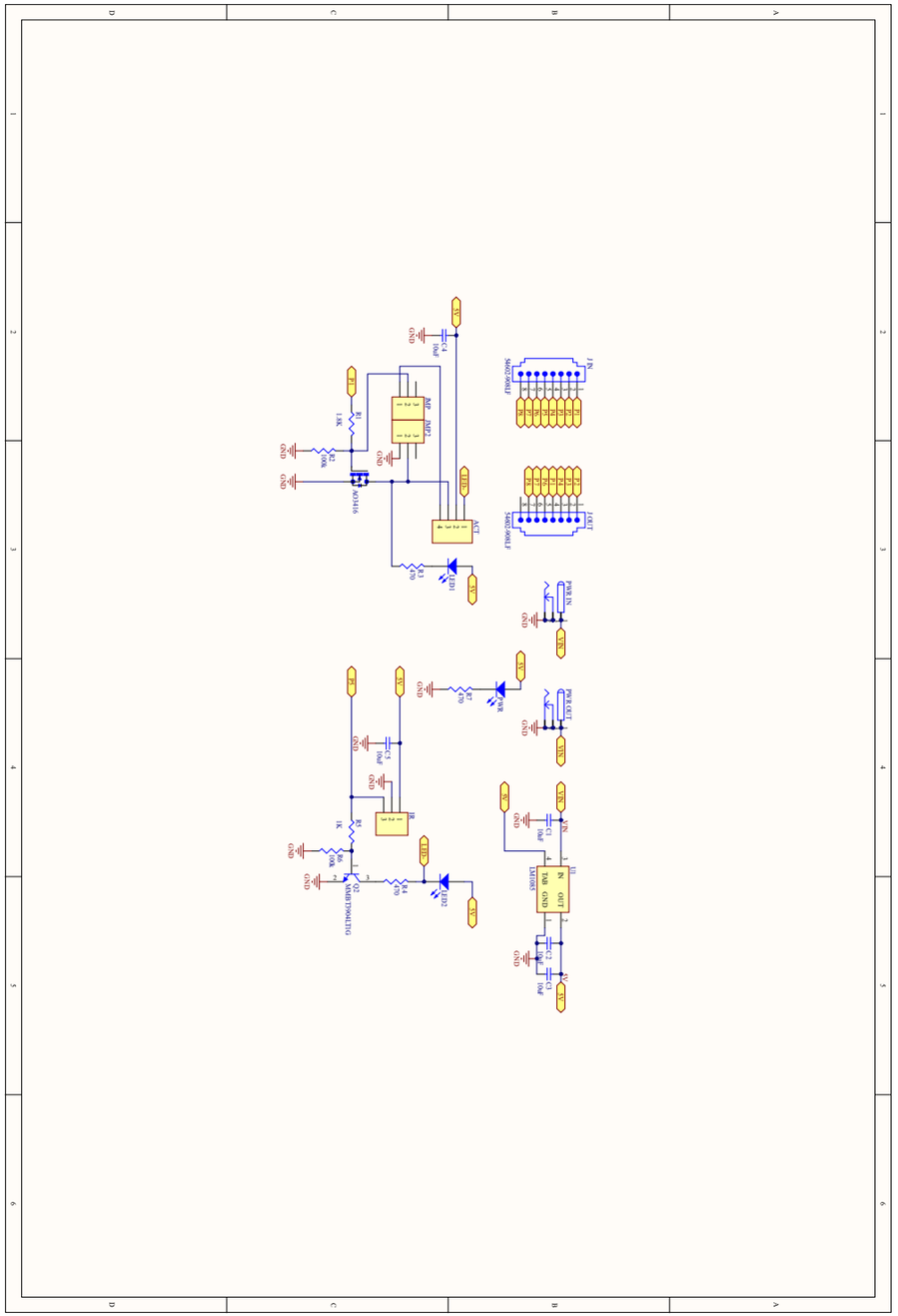


PCB Diagram with Trace Paths

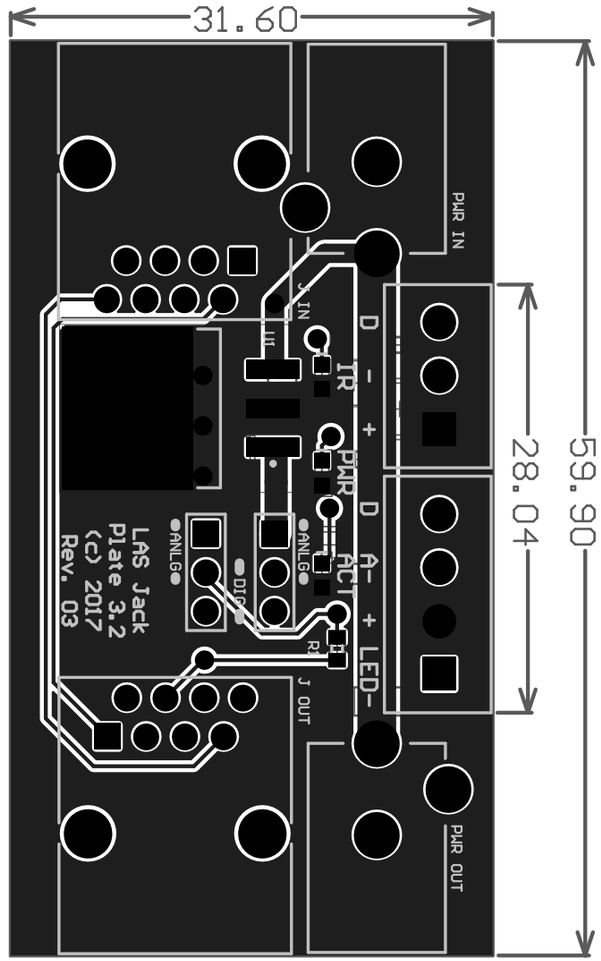


3.2 Jack Plate

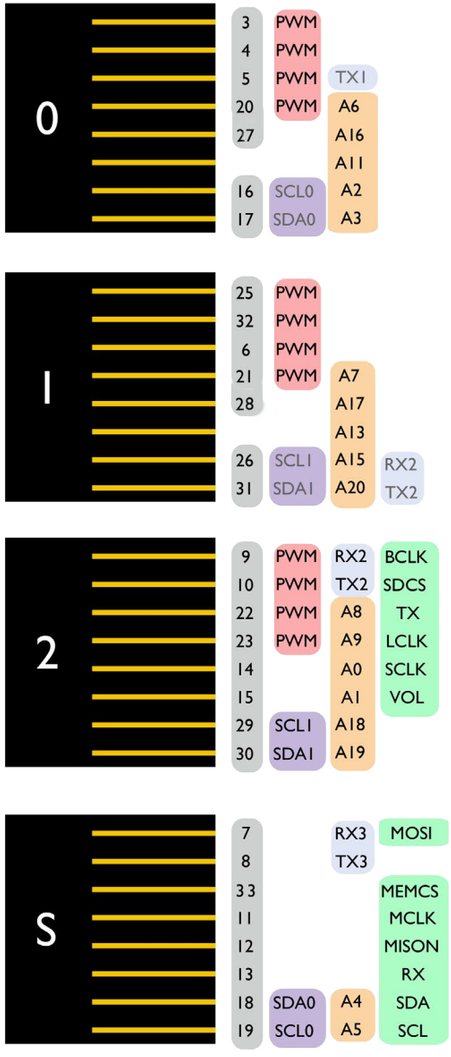
PCB Schematic with I/O Pin Diagram



PCB Diagram with Trace Paths



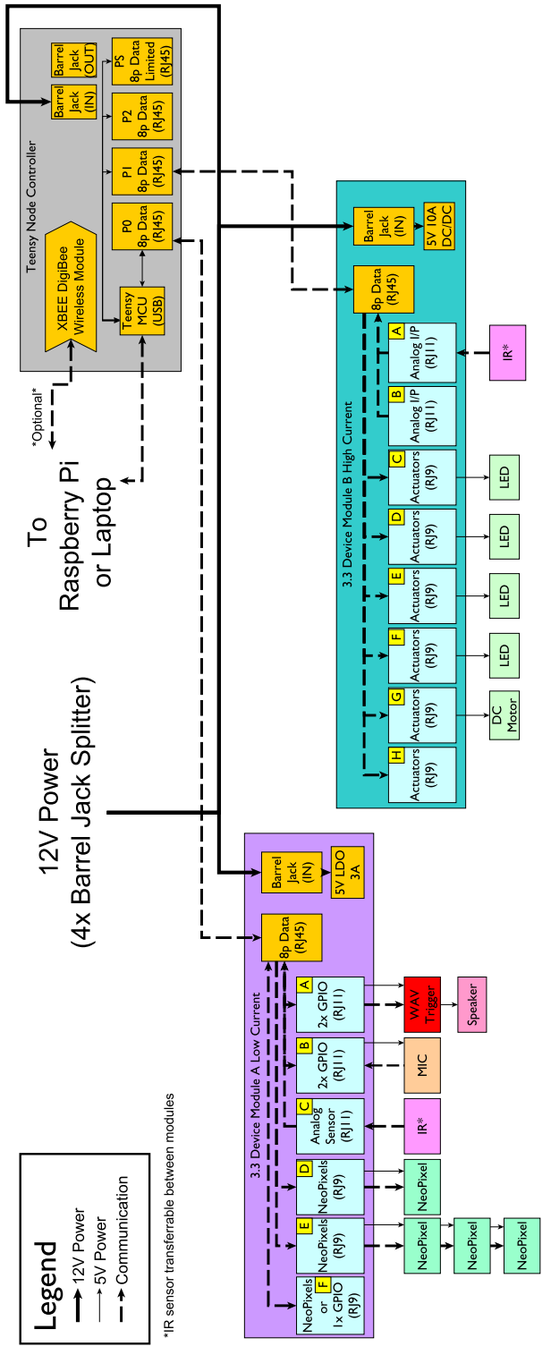
Pinout Diagram for Teensy microcontroller to Node Controller Ports



Note: This diagram is useful to determine the connections between Teensy pins and Node Controller 8P8C modular jack ports - for communication with the device modules. Further documentaion of node ports can be found in the node port library on GitHub (.h file).

Example hookups of peripheral devices

BLOCK DIAGRAM



SETUP

The Desktop Kit is an interactive system that contains a set of distributed hardware and embedded software. These hardware and software systems work together to formulate and generate behaviour on a small scale. Information about the hardware can be found in the *Device Descriptions and Device Diagrams* sections of this manual. Information on how various components are connected can be found in the *Interactive System Network Diagrams*. Instructions to access the different components of the hardware, as well as additional documentation necessary to create your own manipulations of the hardware can be found on the USB stick provided or on the GitHub code repository.

Software to Download

Application	Version	Download Link	Other Notes
Arduino IDE	1.8.5	https://www.arduino.cc/en/Main/OldSoftwareReleases	Download to the local Documents folder on your computer.*
Teensyduino	1.40	https://www.pjrc.com/teensy/td_download.html	Follow install instructions on website. Ensure Teensyduino files are located in same location as Arduino files on computer.
NoMachine	5.3.12	https://www.nomachine.com/	Remote desktop communication with Raspberry Pi.
Python (IDLE)	3.6.3	https://www.python.org/downloads/	Code Raspberry Pi.
TyTools	N/A	http://neodd.com/tytools#overview	Download from GitHub (link on site), instructions on download link. Used to communicate to specific Teensy, from Raspberry Pi.
Platform IO	N/A	http://platformio.org/platformio-ide	Install for Atom. Used for coding in C/C++.

*When uploading C code to your Teensy, off an Ardinuo INO sketch, ensure the correct board is chosen and the correct COM port is chosen. Check which COM port your Teensy is connected to by checking USB ports in Device Manager (on Windows). You can change the board selection to "Teensy 3.1/3.2" on the Arduino IDE by going to Tools -> Board:"".

To learn more about the applications and the code you will be playing around with, make sure to check out information and user manuals created by the developers of these applications. Situations where you may require help from external sources include use of the Raspberry Pi.

Writing Customized Software for Sentient Architecture

Pre-written libraries are included and evolving on the related GitHub pages, however it is your choice to use some or even none of them. To create custom software for your Desktop Kit you will need working knowledge of

- Arduino/C++ (this code runs on the node controller)
- Python (this code runs on the Raspberry Pi or laptop)
- Reading PCB schematics

Software Overview

Loaded on the included USB key are libraries (header and .cpp files), examples and documents which can help you understand the software you may need to write to envision your experiments. Ensure to carefully read the comments within each coding file, as well as the README files within each folder, so that you are able to easily compile and understand the purpose of each file.

The libraries are useful for the configuration of the device modules, the peripheral devices (i.e. WAV Trigger, Adafruit NeoPixels, MIC Sensor) and the serial communication between your Raspberry Pi and the Node Controller / Teensy microcontroller. If you refer to the examples within the src folder, the comments take you through a step-by-step introduction to the use of the IR sensor and the MIC sensor. These examples do not use the libraries; however, when you set up your own interactive system, you can replicate the examples by using the " #include "filename.h" " function at the start of your Arduino sketch. This can allow you to reference functions within your libraries, during your research experiments. All the libraries are located within the lib folder of the code package.

A good start to learning about the system would be to go through the setup detailed in the examples of the src folder.
The final important folder located within the code package is the docs folder, which includes setup for your XBee (an optional device to include within your system), the pinout diagram for the Teensy and a document detailing the software scope and codes for communication between the Teensy and Raspberry Pi. If you so wish to enable this communication, please read through this document thoroughly for your understanding. There is a configuration header file within the lib folder, which defines each one of these codes as well.

Software Included on USB Key

Below is a list of the files and folders included on your USB stick, along with short descriptions of the application of these files. Please refer to the README files within each folder, as well as the comments within each programming file, to gain a comprehensive understanding of each file. You can open all files in PlatformIO, following the configuration settings on the USB key.
The USB key includes three main folders, along with the sub-folders listed:

"docs"

- Series 3.2 Node Pin Mapping - *Describes Teensy pin functions and assignments to Node Controller ports*
- → Software Scope and Communication Protocol - *Communication protocol between Teensy and Raspberry Pi*
- → XBee Module Wi-Fi Setup Guide - *Setting up XBee Module on Node Controller*
- →

"lib"

- →
- → Adafruit_NeoPixel - *For NeoPixel communication and testing*
- → config - *Configuration of peripheral devices for Raspberry Pi communication*
- → Desktop_Kit_Pi - *Raspberry Pi Python code*
- → dm_high_current - *High Current Device Module control*
- → dm_low_current - *Low Current Device Module control*
- → node_ports Relating - *Node Controller ports to Device Modules*
- → SoftPWM - *Software PWM library for non-PWM Teensy pins*
- → SoftwareSerial - *Serial communication between Teensy and Raspberry Pi*
- → sound_detector - *Library for MIC Sensor control*
- → usb_serial_comms - *USB serial communication for Raspberry Pi communication to Teensy*
- → wav_trigger - *Library for WAV trigger control*
- xbee_to_serial - *Library for XBee to serial communication*

"src"

- → node_controller - *Example serial communication between Teensy and Raspberry Pi when WAV Trigger connected*
- simple examples
  - ir_led - *Example of IR sensor communication with LED on Cell Breakout Board*
  - wav_mic - *Example of WAV trigger and MIC Sensor control*

Software Examples

Examples of code written from scratch, using none of the pre-written libraries, can be found at the following link. In particular, they will detail how to determine the correct pin mapping of your hardware setup.

https://github.com/pbarch/1712-Series-3.2-Desktop-Electronics-Kit/tree/develop/simple%20examples

Useful Links

Teensy Pinout Description: *https://www.pjrc.com/teensy/pinout.html*
WAV Trigger Library: *https://github.com/robertsonics/WAV-Trigger-Arduino-Serial-Library*
XBee Arduino Library: *https://github.com/andrewrapp/xbee-arduino*
XBee Demo: *https://github.com/liwp/python-xbee-demo*
XBee for Python Development: *https://github.com/nioinnovation/python-xbee*
NeoPixel Library: *https://github.com/adafruit/Adafruit_NeoPixel*

Node Controller & Raspberry Pi Communication

The communication between the Node Controller and the Raspberry Pi is achieved through two methods.

1. Micro USB cable connected to a USB Hub linked to the Raspberry Pi - hardwired connection

2. The Raspberry Pi and Node provide capability for wireless communication using the XBee ZigBee Module.

Cable Type: Micro USB Cable

If you wish to enable wireless communication between your Raspberry Pi and the Node Controller, there will be a few extra steps added to the setup of your kit. Currently, the Node Controller can communicate with the Raspberry Pi or your laptop through a wired connection; however, you can also enable a wireless communication if you wish to explore this concept.

Included within the documents of the code package is a short user manual to set up the XBee Module on your personal computer. The same can be applied to the Raspberry Pi; just ensure your Raspberry Pi is connected to your local internet (Wi-Fi or Ethernet) and you can access the desktop of the Raspberry Pi through the NoMachine software (download instructions included in the user manual). Be sure to use the XBee USB Dongle provided with the kit to set up the XBee Module on the external device of your choice (either the Raspberry Pi or your laptop). The XBee Module must be plugged into the dongle as specified in the short user manual.

Once the setup is complete, you will need to solder the XBee Module to the Node Controller, in the slot labelled "XBee". The pins of the module must go directly through the through-holes, and the module must be positioned to fit the black trace surrounding the holes. You will need a soldering iron and solder to ensure the module is fully connected to the system for serial communication between the Teensy microcontroller and the external device of your choice.

For full completion of the setup of the XBee Module, you will need to provide your own: solder, soldering iron and internet.

Power

The main DC power is supplied by the provided AC/DC Wall Adapter, which can output a maximum of 3 A.

Cable Type: 18 AWG Stranded Cable Assembly with Barrel Jack Ends, Female to Male, Black Cables

Includes: Black Splitter 4-Way, 3' Barrel Extension

These cables run from the power supply (wall adapter) to any device that requires power from a barrel jack. Power can be distributed from the power supply to more components using the barrel splitter. The following components require power from the main DC power supply: Node Controller, Device Module A Low Current, Device Module B High Current and the Jack Plate.

Raspberry Pi + Teensy Communication

The following section of the manual runs through the setup and control, if you wish to use your Raspberry Pi (RPi) as a messenger for communication between several Teensy devices.

1.1 Installing Raspberry Pi OS on First Use

Materials you will need include:
- HDMI Cable
- Monitor (Computer screen only)
- USB power for your RPi
- Keyboard connection to RPi over USB
- Mouse connection to RPi over USB
- SD Card on your RPi
- USB -> SD Card adapter

Follow these steps to easily set up your RPi:

1. Put your SD card into the USB -> SD Card adapter. Go to: https://www.raspberrypi.org/downloads/ . Download the NOOBS software. Unzip NOOBS folder software into the SD Card on your RPi.
2. Insert SD Card back into RPi. Connect keyboard and mouse to USB ports on RPi. Insert HDMI connection into RPi and connect to monitor. Power the RPi over USB. The screen should now show RPi start up.
3. Allow installation of the RPi OS (Raspbian). This should take around 10 minutes.
4. Reboot the RPi once complete, restart the system (re-plug USB power). Now, you can start using the RPi.

1.2 Setting up NoMachine on RPi for Remote Access

In addition to the materials required in section 1.1, you will also need:
-Ethernet connection between the RPi and your local internet

Now, to set up NoMachine on your RPi, ensure your RPi is still connected to the monitor from section 1.1, as well as the mouse and the keyboard. The RPi should be powered. Now:

1. Open the terminal window on the RPi. Type: `sudo raspi-config`
2. For the following steps, you may have to navigate through the options. But, you must "Enable Boot to Desktop".
3. Next, allow for "Desktop Log in as user "pi" at graphTical desktop".
4. Click finish on the window, then reboot the system.
5. Once the system has rebooted, go to: https://www.nomachine.com/download/linux&id=29&s=Raspberry on the Chromium browser. Download "NoMachine for Linux ARMv7 DEB" under the "NoMachine for Raspberry Pi 3" section. This download will work for the Linux version, using Raspbian OS, on your RPi Model 3.
6. Open the location where the NoMachine package has been downloaded in your RPi (most probably your Downloads folder).
7. Open the terminal window in this folder (right-click for option). To install NoMachine on your RPi, type in the terminal window: `sudo dpkg -i nomachine_6.0.66_1_armhf.deb` *If the version of NoMachine you have downloaded does not match "6.0.66_1", type the number that you see for the version you have downloaded. You could also double-click the folder to install the package from the Downloads folder.
8. If you have NoMachine installed on your computer already, go on to your personal computer. You should be able to see a Raspberry Pi ready to connect to your computer over NoMachine. The setup is complete!
9. If there is a prompt to login to the RPi, the username is: pi. The password is: raspberry.

1.3 Installing TyTools TyQt on your RPi for Communication with Teensy

You do not need the monitor or keyboard or mouse for your RPi. Ensure your RPi is powered over USB by your laptop/computer. Also, ensure the Ethernet connection between your RPi is well-connected. Now, you can proceed with this setup:

1. Connect to your RPi Desktop interface over NoMachine on your laptop. Enter the Chromium browser and go to: https://github.com/Koromix/tytools/releases . Scroll down and download the "Source Code (tar.gz)"
2. Unzip the file in your Downloads folder. Now, follow the steps outlined from the GitHub page for Linux setup:

Build on Linux

TyTools can be built with GCC or Clang.

Prerequisites

To install the dependencies on Debian or Ubuntu execute:

```
sudo apt-get install build-essential cmake libudev-dev qtbase5-dev pkg-config
```

On Arch Linux you can do so (as root):

```
pacman -S --needed base-devel cmake udev qt5-base
```

Compilation

Open the project directory in a terminal and execute:

```
# You can of course use another directory if you prefer.
mkdir -p build/linux && cd build/linux
cmake -DCMAKE_INSTALL_PREFIX=/usr/local ../..
make
```

If you want to build debug binaries instead, you should specify the build type:

```
cmake -DCMAKE_BUILD_TYPE=Debug ../..
```

The compiled binaries can be used directly from the build directory. Follow through the next section if you want to install the application.

Installation

You can deploy TyTools to your system with the following commands:

```
sudo make install
```

By default this will copy the files to `/usr/local`. To change this directory you need to change the `CMAKE_INSTALL_PREFIX` value in the Compilation section above.

You do not need to debug binaries, skip that step. Ensure you check the prequisites (you are using Debian) and install them in the terminal window. When opening the project directory, right-click on the folder you unzipped, containing TyTools, and select "Open in Terminal". You can then type the commands, starting with 'mkdir', on separate lines. Once you complete these steps, TyTools has been installed on your computer!

3. Open the TyTools folder in your Downloads folder. Go to *build -> linux.* Scroll down to open **tycommander- er.** You will need to open this executable file as an administrator within the Raspberry Pi, since you will not be able to upload code to the RPi without enabling this condition. In order to accomplish this, you must:
    a. Right-click on the **tycommander** executable file.
    b. Select "Open with…" option.
    c. A window will appear. Switch to the second tab of the window, the "Command Line".
    d. In the box, type: sudo . Then, press "Enter". The TyQt user interface will open, with admin privileges (i.e. uploading code to attached Teensy devices).
4. Your setup of the TyTools application is complete! Some tips to help you to navigate through the user interface:
    a. You can see the Teensy ID of the Teensy you have connected to your RPi in the left margin of the interface. In the *Information* tab of each respective Teensy you have connected, you can see the "Path", identifying the USB port that is being used by the Teensy for communication with the RPi.
    b. In the *Serial* tab, you can see the Serial output of your Teensy device, if you put statements such as "Serial.println("")" within your Arduino .ino sketch.
    c. The *Options* tab allows you to upload code to your Teensy, through the RPi, instead of having to do it through an Arduino script. This will be explored in the next section. You can also edit the serial output in the previous tab.
    d. Use the *Upload* button to upload code to the Teensy. Use the *Reset* button to reset the Teensy and restart a program on the Teensy. The *Bootloader* button is to be used only when you are using a Teensy for the first time. You probably do not need to use this button during programming. Finally, the *Serial* button allows you to enable serial output on Serial terminal in TyQt. This will be useful when communicating with your RPi.
    e. For further details about the TyQt user interface, visit: http://neodd.com/tytools .

1.4 Uploading Code from Arduino Sketch to Teensy Connected to Raspberry Pi

With TyQt installed, you do not need to use the Arduino app available on the RPi. If you like coding on your laptop, which is probably faster than the Raspberry Pi, you can upload code to your Teensy through the TyQt user interface. This section will walk through the steps on how to achieve this, assuming you have a completed Arduino sketch on your laptop / personal computer. Also, it is assumed your Raspberry Pi is connected to NoMachine, and you can access all its features on its desktop interface.

1. Within your Arduino application, go to *File -> Preferences*. There will be an option in the window that appears to "Show verbose output during …" Beside this statement are two boxes. Select the "compilation" option. You can now apply the changes and exit the window.
2. On your Arduino sketch, *Verify* your code. In the compilation screen below your script, a lot of processes will appear. This means that the verbose output is working.
3. Once the verification of the code is complete, you must look for a .hex file created by the script. You can find the file path in the compilation output, around 4-6 lines from the bottom of the compilation output. You may have to scroll to the right. The file location will end in ".ino.hex". Select and copy the file location.
4. Open File Explorer on your Windows computer (or equivalent for Mac / Linux). Paste the file location, and the .hex file will open. Copy the entire .hex file onto your clipboard.
5. Go to your Raspberry Pi desktop. Go to "Documents" and right-click to "Open Empty File".
6. Name the file the same name as the name you gave the Arduino sketch. IMPORTANT: Make sure you end the file name with ".hex". This ensures the file stays as a .hex file, and does not become a .txt file.

7. Paste the .hex code from your personal computer into the newly created file. Save, then exit the file.
8. Open TyTools, using the same steps identified in section 1.3 (after the installation process). Remember to enable admin privileges!
9. Select the Teensy which you will be uploading code to, in the left margin. Go to the Options tab, and within the Firmware: box, select Browse. Go to the location of the .hex file, and select it.
10. Select the *Upload* button, and the Teensy should reset, then load the code. A bar indicating the percentage of code being uploaded should be displayed below the Teensy ID in the left margin, during the upload pro cess. You have now completed the process of uploading code to your Teensy, and the program should be running!

1.5 Reading Data from your Teensy on the Raspberry Pi

Say you would like to perform an operation on your Raspberry Pi, to read certain outputs from your Teensy. In this section, a simple start to outputting data from your Teensy on the RPi terminal is presented. It is up to you on how you wish to expand this example.

1. You must have some sort of serial communication enabled in your Arduino sketch (e.g. *Serial.write(), Serial. read(), Serial.print(),* etc.)
2. You will need some code from your USB. On your USB, go to the src folder, then open the python file for "Simple communication between the Teensy and Raspberry Pi".
3. Open the file. The path of the Teensy in the code may be different than the path of the Teensy connected to your RPi. Make sure to change that string to suit the connection to your RPi.
4. Copy the file, and paste the file within the pi folder on your RPi.
5. Disable serial communication with TyQt by disabling the *Serial* button on the TyTools user interface. Serial communication works like a mutex; only one process at a time can have the key to access serial communication.
6. Open the Terminal Window on your RPi. Enter: `python serialComm.py` . Your serial communication is now enabled with your RPi. Now, it is up to you on how you wish to alter either the Arduino sketch or the python code, to parse through the data!

Data/Pinouts

For pinout control and data reading between the Node Controller and peripheral boards such as the Device Modules and Jack Plate, an RJ45 cable is used to facilitate that. These cables will run between the Data ports of the Node and the Data ports of the peripheral boards.

Cable Type: 26 AWG Stranded, flat 8-wire bundle with RJ45 ends, White

Peripheral Control

For manipulation of sensors and actuators, PTSM Phoenix Contact and RJ9 cables are used between peripheral boards and their peripherals.

IR Sensor Cable, WAV Trigger Cable, MIC Sensor Cable

-Cable Type: 26AWG Stranded, flat 6-wire bundle with RJ11 ends, White

Cell Breakout Cable, NeoPixel Cable

-Cable Type: 26 AWG Stranded, flat 4-wire bundle with RJ9 ends, White

Node Controller Ports and Associated Teensy Pins for Communication with Device Modules

Node Controller Ports	Associated Teensy Pins (Digital)	Associated Analog Pins (from Teensy microcontroller)	Low Current Device Module Associated Ports	High Current Device Module Associated Ports	Jack Plate Associated Ports (Screw Terminal)**
P0	3		A	C	ACTUATOR
	4		A	D	
	5		F	E	
	20	A6	E	F	
	27	A16	D	A	IR
		A11	C	B	
	16	A2	B	G	
	17	A3	B	H	
P1	25		A	C	ACTUATOR
	32		A	D	
	6		F	E	
	21	A7	E	F	
	28	A17	D	A	IR
		A13	C	B	
	26	A15	B	G	
	31	A20	B	H	
P2	9		A	C	ACTUATOR
	10		A	D	
	22	A8	F	E	
	23	A9	E	F	
	14	A0	D	A	IR
	15	A1	C	B	
	29	A18	B	G	
	30	A19	B	H	
PS	7		A	C	ACTUATOR
	8		A	D	
	33		F	E	
	11		E	F	
	12		D	A	IR
	13		C	B	
	18	A4	B	G	
	19	A5	B	H	

This table provides a user-friendly representation for how each pin within the Teensy microcontroller can control your setup. Though you could use the node_ports.h library to help you determine which pins activate select ports on your device modules, this table makes it easier to code your Arduino from scratch.
For example, if you connect an IR sensor to port 'B' on your Low Current Device Module, you can determine which pins to activate in your Arduino program in order to read sensor data from the IR sensor.

In this case, if you connected your Low Current Device Module to port P1, you can either activate pins 26 and 31 to be able to read the sensor data. If you connected the Low Current Device Module to P0, you can either activate pins 16 or 17 to read the sensor data.

The same works for the High Current Device Module. If you decide to connect your Cell Breakout Board (including the LED) to port 'E' on this module, and your High Current Device Module is connected to port P2 on the Node Controller, then pin 22 of the Teensy can be used to activate a PWM signal to turn the LED on/off.

This table is essentially bridging a connection between the schematic diagrams of the device modules, and the Teensy pin diagram. Through an 8P connection between the Teensy and any device module, it is simple to trace which Teensy pin can activate the selected port on your device module. However, the Teensy pin diagram is helpful to understand the capabilities of each pin (e.g. pin 16 is a clock, pin 17 can be used for data transfer, etc.).

The table also lists the associated analog pins of the Teensy, while the Teensy pin diagram lists the pins which have hardware capabilities. For example, if a pin does not have hardware PWM capabilities, as noted by the "PWM" sign next to select pins, you may need to implement a software PWM to impose attributes such as a duty cycle. The software PWM library is included in the "lib" folder of your USB Key. Ensure to #include "SoftPWM.h" in your Arduino sketch when necessary.

Examples of how this is applied are available in the src folder of your USB Key. Applications of the WAV Trigger, IR Sensor and MIC Sensor are available, within the context presented.

****Extra Note for Jack Plate Modules**

If you have more than one Jack Plate module in your kit, you have the capability of "daisy-chaining" your Jack Plate modules. This means that you do not need to connect each Jack Plate to an 8P port on your Node Controller. Instead, you can connect all your Jack Plates (up to a maximum of 4) through the 8P "J OUT" connections. Therefore, one Jack Plate must be connected through the "J IN" port, to your Node Controller. However, the next Jack Plate can be connected to the "J OUT" port of the first Jack Plate. This connection would be made to the "J IN" port of the second Jack Plate. This pattern would continue to the last Jack Plate.

If you wish to control actuators or LEDs on the "daisy-chained" Jack Plates, you will have to activate different pins than those activated for the first Jack Plate. Below is a table identifying which pins you would have to activate, depending upon the Node Controller ports, for the "daisy-chained" Jack Plates (refer to the connections of the first Jack Plate in the chain in the first table).

The IR sensor is indicated as a possible connection to the Jack Plate. However, the IR sensors included in the Desktop Kit included on their own board, which does not need to be connected to the Jack Plate, but instead a low or high current device module. The IR sensor screw terminal connection is still listed in the table for completion purposes, in relation the Jack Plate schematic. However, if you wish to communicate between an actuator (such as an SMA) on the Jack Plate, and an IR sensor, you can follow the following steps:

1. Connect your Jack Plate to one of the Node Controller ports.

2. Connect your Low Current or High Current Device Module to another Jack Plate, with an IR sensor connected to the device module being used in the setup.

3. Ensure the IR sensor is able to read data (e.g. pinMode(pinNumber, INPUT) then analogRead(pinNumber)), while the actuator on the Jack Plate is set up as an output connection, with an analogWrite function in the void loop().

These steps would ensure communication between the IR sensor and an actuator on the high-power Jack Plate.

Node Controller Ports and Associated Teensy Pins for Communication with Jack Plates

Node Controller Ports	Associated Teensy Pins (Digital)	Associated Analog Pins (from Teensy microcontroller)	Jack Plate #2	Jack Plate #3	Jack Plate #4
P0	3				
	4		ACTUATOR		
	5			ACTUATOR	
	20	A6			ACTUATOR
	27	A16			
		A11	IR		
	16	A2		IR	
	17	A3			IR
P1	25				
	32		ACTUATOR		
	6			ACTUATOR	
	21	A7			ACTUATOR
	28	A17			
		A13	IR		
	26	A15		IR	
	31	A20			IR
P2	9				
	10		ACTUATOR		
	22	A8		ACTUATOR	
	23	A9			ACTUATOR
	14	A0			
	15	A1	IR		
	29	A18		IR	
	30	A19			IR
PS	7				
	8		ACTUATOR		
	33			ACTUATOR	
	11				ACTUATOR
	12				
	13		IR		
	18	A4		IR	
	19	A5			IR

References

"Living Architecture LIAR," accessed February 2, 2022, https://livingarchitecture-h2020.eu/.

Lucy Bullivant, *4dsocial: Interactive Design Environments* (London: AD/John Wiley & Sons, 2007)