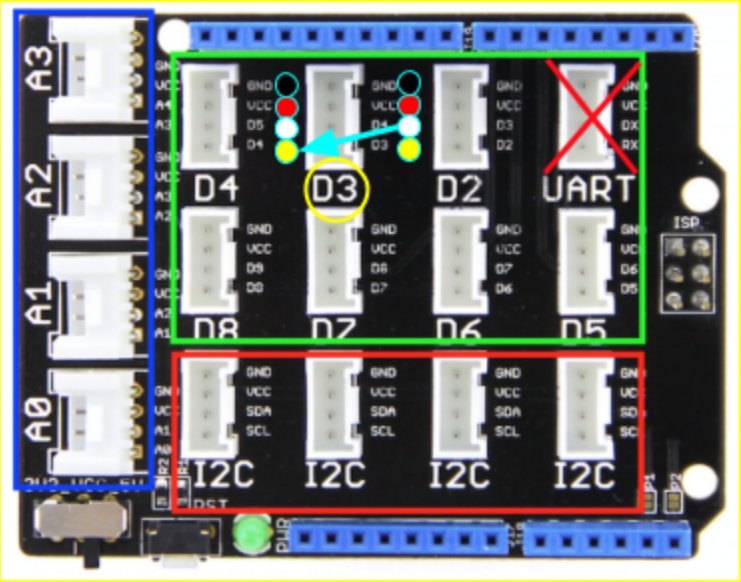
ME 216M: Introduction to the Design of Smart Products

Project 1 Guidelines

Please read this document very carefully before starting your project. It will be updated with corrections/new information throughout the week.

Hardware: Grove Shield



The Grove shield allows you to easily connect modules to the Arduino.

These can be sensors, actuators, or more advanced devices (like the mp3 player), which communicate over I2C.

The shield has labeled connections to the analog and digital pins on the Arduino, as well as, connections to the I2C bus.

Three types of ports:

- Digital GREEN
- Analog BLUE
- I2C RED

The Grove 4-wire connector

The Grove four-wire connector provides:

- Ground BLACK GND
- Power RED VCC
- I/O pin WHITE For example, SIG (analog/digital signal), TX/RX (I2C), NC (not connected), ...
- I/O pin YELLOW For example, SIG (analog/digital signal), TX/RX (I2C), NC (not connected), ...

I/O pins and overlap on connectors!

Simple modules (e.g., button, LED, potentiometers, ...) only use one I/O pin as signal.
You can inspect these modules and note that YELLOW pin = SIG and WHITE pin = NC (not connected).

If you use a module that uses two signal pins simultaneously, they cannot be plugged next to another module. Their connector will occupy both the port and an adjacent port. The figure shows how the D3 port's YELLOW pin connects to YELLOW, while WHITE pin is connected to pin D4 (which would be the YELLOW pin for the adjacent D4 port). Unless the WHITE pin is NC, it will interfere with the D4 port.

Do not attach anything to UART

It is used for USB-over-serial communication, for programming and debugging. Must be unplugged!

Mp3 boards: SoftwareSerial Library can be used on any digital pins (green)

The MP3 boards communicate over a serial link. The SoftwareSerial library allows us to use any digital pins for serial communication. More info, below (Software).

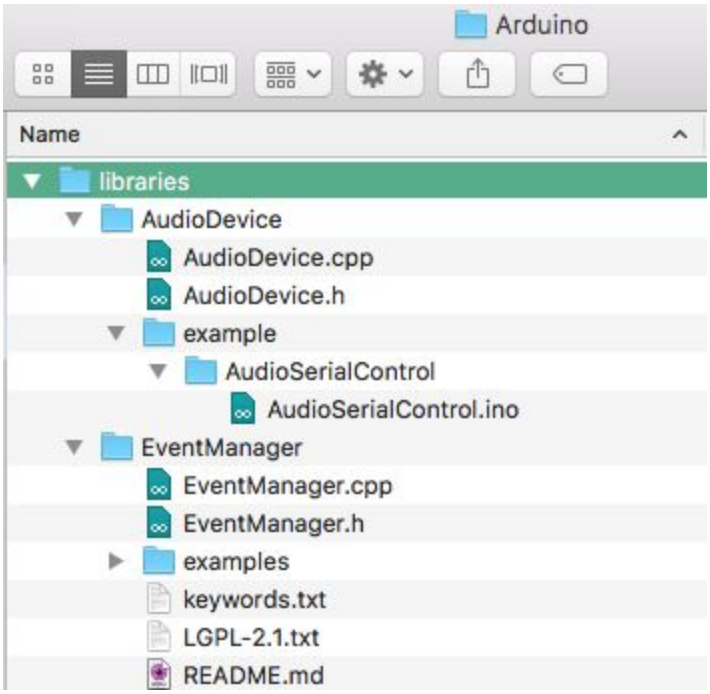
Radio: Use I2C ports (red)

If you are using the FM radio, connect it to I2C ports. I2C is a master-slave communication protocol, where multiple slave devices can share the same bus and master. You can connect additional I2C devices to the I2C bus, but you cannot use these connectors for other functions.

Software: Arduino

There are several steps you need to complete in order to communicate with the MP3 board.

There are several steps you need to complete in order to communicate with the MP3 board(s) or the FM radio. In addition to these, make sure you read the documentation for the Lab2 class found in the AudioDevice.h file.



Installing libraries

- Download the library
- Place it in the libraries folder in your Arduino sketchbook
- Restart the Arduino IDE to refresh the library list

AudioDevice Library

- Download and install the AudioDevice library from canvas

EventManager Library

- Download and install the EventManager library from GitHub: https://goo.gl/lHe6iB
- Read the README on the GitHub page
- Restart the Arduino IDE to use it

[Optional] IMU libraries

- Download the Grove IMU and I2CDev libraries if you want to use the IMU.
- https://github.com/Seeed-Studio/Grove_IMU_9DOF

Using the libraries for your Arduino sketch

1. Restart the Arduino IDE to refresh the library list (if you haven't already)
2. Test the AudioSerialControl example
3. Read the AudioDevice.h
4. Make sure to include

```
#include <AudioDevice.h>
#include <Wire.h>
#include <SoftwareSerial.h>
#include <EventManager.h>
```

5. Mp3 players

   Instantiate the mp3 player like this:

```
const int PIN_MP3_TX = 6;
const int PIN_MP3_RX = 7;

AudioDevice mp3(PIN_MP3_TX, PIN_MP3_RX, mp3b); //mp3a or mp3b
```

   The constructor takes three arguments. The first two are only used for the mp3 players. In this case, these are the pin numbers of the RX and TX pins used by the SoftwareSerial Library.

   Keep in mind that mp3 TX is connected to Grove Shield/Arduino RX and vice versa:

mp3 TX ↔ Grove Shield RX
mp3 RX ↔ Grove Shield TX

mp3a device: http://wiki.seeed.cc/Grove-Serial_MP3_Player/

mp3b device: http://wiki.seeed.cc/Grove-MP3_v2.0/

6. FM radio

   Instantiate the FM radio like this.

   ```
   const int NOT_USED = -1;

   AudioDevice fm(NOT_USED, NOT_USED, fm); //the first two arguments are ignored for FM radio
   ```

7. Make sure you call

   `initHardware()`

   in setup(), with the rest of your hardware initialization.

EventManager for State Machines

First, read the github page on the EventManager(https://goo.gl/lHe6iB) for context.

You can think of every listener function as a state machine.

Whenever an event is posted, it will be passed to each listener connected to that event. Events are processed when the member function processEvents() is called on an EventManager.

An example of how to use the EventManager class to create state machines can be found in the Arduino sketch EventManagerEx.ino. Each state machine uses a switch statement to perform different things based on what the current state of the state machine is. In each case, the state machine can look for specific events and parameters to perform different code or to change to a different state.

You can upload this sketch to an Arduino and use the Serial Monitor to see the state machines toggling between different states through different events.

Miscellaneous

1. If you are using the FM radio and `next()` and `previous()` do not seem to be working, try going outside or closer to a window. During testing, I had some trouble getting a strong enough signal from anything other than KZSU while on the bottom floor of building 550.
2. If you are still experiencing issues, you can try changing the #defined constants in the AudioDevice.cpp file:
   - `FM_NUM_CONNECT_ATTEMPTS` defines the number of times the FM radio attempts to verify that the frequency it is on is a valid station
   - `FM_MIN_SIGNAL_STRENGTH` defines the minimum signal strength for a frequency to be considered valid. Try lowering the signal strength or raising the number of connection attempts.
3. The AudioDevice class is based off examples for the two mp3 players and FM radio. There is additional functionality that we did not include, but that you are welcome to integrate into your project by editing AudioDevice.h and AudioDevice.cpp.
4. You should instantiate an EventManager object to have a global scope. If needed, you can increase the number of Listeners and the Queue Size by following the directions on the GitHub page.