0 - Directions

Every time you start learning a new language or start a new project you will be expected to create a new Project Description Page.

How to setup a Project Description:
1. Add a new page and start it with the next available page number followed by a hyphen and a description of the what you are working on.
2. On the top left of the sheet list your period, team members and what languages you will be learning / using.
3. Write a well worded description of what you plan to learn or build.
4. If you are building a game or program.
    a. List out everything your program will be able to accomplish
    b. How input will be handled
    c. How does your program store data?
    d. If you have a game
        i. How does every object in your game behave?
        ii. What does the player do?
        iii. How are levels played / beat and stored?
        iv. Anything that would allow you to fully describe how your game plays
5. Sketches of GUI plans, level layouts or other visual elements.
6. Anything else you need to fully what you are working on.

Rover-X Studying

Period: 6
Team Members: ME
Languages Used: Arduino (C++), java for android

I will be using the Sunfounder's Super Kit v2.0 in order to learn Arduino/C++, available here:
https://www.sunfounder.com/learn/category/Super-Kit-V2-0-for-Arduino.html

Rover-X Plan

I'll be working using an Arduino to create a rover that is controlled by a phone app. I will create and code these things by myself, using the Arduino C++ Language and Java for Android

Directions

On the last day of every week you will update your task list and create a new task list.

Note: I will add to your adjust your list if it is too light or too ambitious.

Updating a Task List:
- New tasks need to be added to the task as they are assigned to members and a target complete date set.
  - ○ **You are never allowed to adjust the target completion Dates!**
- Color all tasks based on the target completion date and status
  - ○ Completed Tasks
    - ▪ highlighted in Green
  - ○ Incomplete Tasks
    - ▪ No highlight if the target completion date has not elapsed
    - ▪ Yellow as soon as the target completion date is missed
    - ▪ Orange the task is 1 week behind schedule
    - ▪ Red the task is 2 or more weeks behind schedule

Creating a Task List for a new week:
- Start a new task list and with the created on date as the Thursday/Friday you filling it out on.
- Move all ongoing or behind tasks to the new list
  - ○ **Do not modify the target complete dates!**
  - ○ **Wait to color task until the end of the week.**
- Add new tasks that your team will be working on

Create on __/__/____

Assigned Team Members	Assigned On	Target Completion Date	Description of the task	Completed on

1st Six Weeks

Thursday, August 8, 2019   12:59 PM

Create on __8/16/19_

Assigned Team Members	Assigned On	Target Completion Date	Description of the task	Completed on
Varun Kumar	8/16	8/23	Complete the first three tasks of the Sunfounder Arduino kit, available here: https://www.sunfounder.com/learn/category/Super-Kit-V2-0-for-Arduino.html	8/21

Create on _8/26/19_

Assigned Team Members	Assigned On	Target Completion Date	Description of the task	Completed on
Varun Kumar	8/26	8/30	Finish Morse Code and Learn to Control Motors	Kinda on 8/30

Create on _8/30/19__

Assigned Team Members	Assigned On	Target Completion Date	Description of the task	Completed on
Varun Kumar	8/30	9/6	Learn to control motors and learn about motor drivers	9/6

Create on _9/6/19__

Assigned Team Members	Assigned On	Target Completion Date	Description of the task	Completed on
Me	9/6	9/13	Get motor controller and interact with the arduino and get new ide	9/13

Create on _9/13/19_

Assigned Team Members	Assigned On	Target Completion Date	Description of the task	Completed on
Me	9/13	9/20	Get motor controller and interact with arduino and connect it to Arduino	

Create on _9/20/19__

Assigned Team Members	Assigned On	Target Completion Date	Description of the task	Completed on
Me	9/20	9/27	Get new motor controller and connect to Arduino	

2nd Six Weeks

Thursday, August 8, 2019   12:59 PM

Create on _10/4/19___

Assigned Team Members	Assigned On	Target Completion Date	Description of the task	Completed on
Me	9/13	9/20	Motor Controller Interaction (because I spent time researching before I got the controller.)	10/3

Create on _10/11/19___

Assigned Team Members	Assigned On	Target Completion Date	Description of the task	Completed on
Me	10/4	10/11	Get remote controller to interact with motor	10/11

Create on _____

Assigned Team Members	Assigned On	Target Completion Date	Description of the task	Completed on
Me	10/11	10/18	Get a radio control module and make a remote control	

Create on _9/6/19__

Assigned Team Members	Assigned On	Target Completion Date	Description of the task	Completed on

Create on _9/13/19_

Assigned Team Members	Assigned On	Target Completion Date	Description of the task	Completed on

Create on _____

Assigned Team Members	Assigned On	Target Completion Date	Description of the task	Completed on

Directions

Thursday, August 8, 2019   1:47 PM

When learning anything new you will be required to keep digital notes here or upload your paper notes here.

The notes should be broken into meaningful sections. You will need to write down things than Java and new terminology. With that said I expect you to have notes…

UML Class Diagram Notes

Thursday, August 8, 2019   1:00 PM

Create Notes on UML Class Diagrams:

You may go out and find information on your own or use the below link:
https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/

UML Activity Diagram

Friday, August 9, 2019   12:02 PM

Create Notes on UML Activity Diagrams:

You may go out and find information on your own or use the below link:
https://www.lucidchart.com/pages/uml-activity-diagram

Directions

Thursday, August 8, 2019   1:02 PM

Every time you start learning a new project you will be expected to create a planning page. Planning a large project should take you 1 to 3 weeks.

For every project you will need to complete the below plan exercises:
- UML Class Diagram
- UML Activity Diagram
- GUI Layout sketches (For GUI based programs)

0 - Project Plan

Thursday, August 8, 2019   1:52 PM

Directions

Every week you will need to create a page with the title being the first date of the week. **Even if there is nothing to upload!**

During learning weeks you likely will not have much to upload here, other than simple programs you create while learning.

IL COMMENCE!!

Project one: Blinking Light- check Vid underneath

Blinking_LE
D
Button Light- check Vid underneath

20190821_
131407

Morse Code!!!

20190828_
132355

Command Learning:

pinMode(#,OUTPUT);- This command is for outputting power to the pin on the Arduino.

The number is written on the Arduino board itself. It is next to its corresponding pin #.

digitalWrite(#,HIGH);- This is the way the LED turns on.

The Arduino powers the slot, and the light that's connected to that slot turns on.
In order to turn the light back off, change the setting to LOW

In order to output to the console/user interface, use the Serial command

Serial.print("");
Serial.println("");

In order to read the user's input, use the Serial.read(). The problem arises when you have to read a String. In order to do this, a char array must be constructed, to collect all the characters of the String. I still haven't figured this out, but I will soon!

BEFORE YOU DO ANY OF THIS, YOU MUST MAKE THE SERIAL AVAILABLE. In your setup() method, you have to "activate" the Serial by using:

Serial.begin(9600);

The 9600 is the speed the Serial Monitor (the User Interface) collects the data. As a default, it is set automatically at 9600 baud.

To access the Serial output, use the Serial.available() command. Using this checks the bytes of data gained in the Serial monitor. If Serial.available() is greater than 0, new data has arrived.

The delay(#) command is much like the Thread.sleep() command in Java. This stops the program from doing anything for the amount of milliseconds specified.

USING CLASSES IN ARDUINO

It's quite simple: use multiple classes in one file. In that way, you can call different attributes of each program. An example program is shown here.

This is a sample program to work with LEDs.

You must still have the setup() and loop() methods.

INTERFACING WITH THE MOTOR SHIELD (v2)

In order to connect to the new Motor Shield via an IDE, these are the steps:

1.)

In order to interface with the motor controller, you must include the Adafruit libraries. This enables the computer to control the functions of the part.

2.)

Now you've included ALL the libraries. The problem is, Adafruit has many Arduino products. In order to access your motorshield, enter the proper command to initialize the Adafruit motorshield object.

3.)

Great! You've connected to your board. Now you need to access the motor on the board to control it. In order to do this, enter the preceding command, making sure to add the asterisk before the variable name declaration. On the other side of the equal sign, make


HARDWARE SELECTION:

OBSOLETE

The motor controller for our ultimate project
is called the L293D. This is a little chip, about 1 inch by 1/2 inch. On each side of the rectangle that makes up the chip, there are 8 prongs that all plug into the breadboard. Each of the boards can support two motors. The ultimate piece of equipment that's going on the rover is called the L293D motor shield.

The L293D motor shield, in all of its beauty-
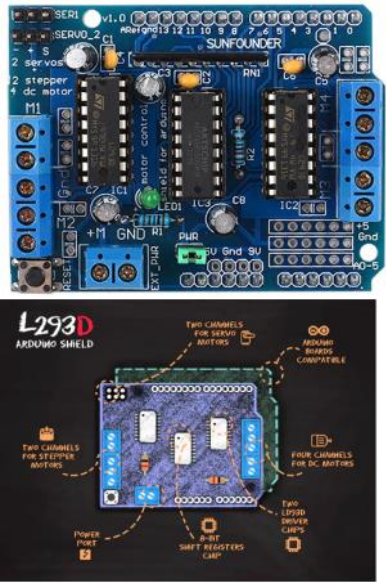
On each pin of the L293D, a different function is used. They are illustrated on the diagram below:
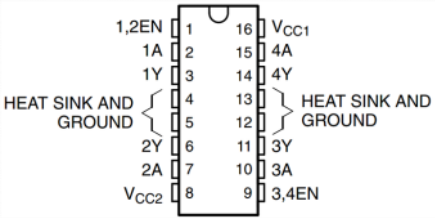
Pin	Name	Function
1	Enable 1,2	Enable pin to control 1,2 driver
2	Input 1A	Input to control 1Y
3	Output 1Y	Output, connect to motor
4	GND	Ground and heat sink
5	GND	Ground and heat sink
6	Output 2Y	Output, connect to motor
7	Input 2A	Input to control 2Y
8	Vcc2	Output supply voltage
9	Enable3,4	Enable pin to control 3,4 driver
10	Input 3A	Input to control 3Y
11	Output 3Y	Output, connect to motor
12	GND	Ground and heat sink
13	GND	Ground and heat sink
14	Output 4Y	Output, connect to motor
15	Input 4A	Input to control 4Y
16	Vcc1	Supply voltage (7 max)

The L293D Motor Shield is quite important in controlling motors, because it combines two L293Ds in one device that can be used to control 4 different motors at different speeds. In order to use the L293D shield, you just attach it to the pins. **This inhibits the addition of another feature, as it takes up all the ports in which features could reside. I'll have to get more Arduinos and add them to the rovers when I add more objects, if that does happen.**

An image to describe the way that a motor controller works. This is a controller from adafruit, and it works the same as the L293D. As is visible, an Arduino is required to control the motors.

SERVO:

For the project, we'll end up needing a servo for correcting the balance of the vehicle when we theoretically put the arm on it. It has a high torque so can make quick adjustments if necessary.

GYROSCOPE:

Way to ensure the rover remains straight. The Arduino, by default, doesn't have a gyroscope previously installed. In order to make the rover stable, the gyroscope will have to make sure that it stays flat. The gyroscope is going to be external, so the ports will have to be open in order for

Great! You've connected to your board. Now you need to access the motor on the board to control it. In order to do this, enter the preceding command, making sure to add the asterisk before the variable name declaration. On the other side of the equal sign, make sure to use your MotorShield object that you declared in Step 2. In order to get a motor, use the getMotor() command, making sure to put your port number in the parentheses. The port numbers are in front of the "m"s, which are on the left

4.)

In your setup() function, call begin() on the Adafruit_MotorShield object:

Like the image said, in the setup function, call the begin method.

5.)

This is the fascinating thing about the MotorShield v2. In order to run the motor, all you have to do is tell the motor to run(direction) [forward, backward]

6.)

You can set the speed of the motor using this command. The greater the number, the greater the speed. When you set the speed of the motor to 0, it stops.

DESIGN PROCESS:

*if a word is underlined, I don't know if the term is correct and will find out

1. The first step is to get a remote-controlled car.
2. Once we have the remote controlled car, we have to remove the chassis of the vehicle (the little plastic shell)
3. This next step is quite complicated. We will dismantle the car so we can add *our* board to it.
   a. The first step is to find the mainboard of the car, and other components attached to it that aren't motors.
   b. Remove the mainboard and other components until all that you're left with is a chassis and motors.
4. In this way, we reduce the time spent building and increase the time programming. However, *some* building will be required.
5. Inevitably, we'll have to attach the motors to our L293D shield. We'll use the crimp connectors on the board to connect to the ports. This is our forward propulsion.
6. That's good and all, but any good rover needs steering. The good thing, however, is that the RC car has a built in steering system. We'll need to examine the system and connect the servo to the board. The good thing about the L293D shield is that it has two places, one on each side, where you can plug in the wire.
7. Once you've connected all the crucial parts, begin to program. Yay me.

For the project, we'll end up needing a servo for correcting the balance of the vehicle when we theoretically put the arm on it. It has a high torque so can make quick adjustments if necessary.

GYROSCOPE:

Way to ensure the rover remains straight. The Arduino, by default, doesn't have a gyroscope previously installed. In order to make the rover stable, the gyroscope will have to make sure that it stays flat. The gyroscope is going to be external, so the ports will have to be open in order for it to work.

ALTERNATE MOTOR SHIELD (because Adafruit updated the old one…ugh)

I can't believe that I actually have to do this, but here I am.

→ Because of the fact that I have to design the circuit before I build it, I'm using a software to build the circuit. This software doesn't have the L293D shield, because the makers of the shield, Adafruit, decided to upgrade it. To create the part would cost like $30, so nah…

The part is called the TB6612 motor shield. Here it is:

Luckily enough, the part still connects in the same way to the Arduino: via the little plugs connected to the pinholes in the Arduino.

Yay…

Man I'm really frustrated so woohoo

The formal term is drop-in compatible.

Edit- 9/23/19: I change my mind. This is awesome!

Here's how it works. The blue crimp connector on the side that's only two holes wide is where the DC Power goes. You have to put batteries connected there. It is possible, however, connect it via the Arduino's DC Power supply. More on that later…

This is the same as the old one, luckily enough.

fritzing
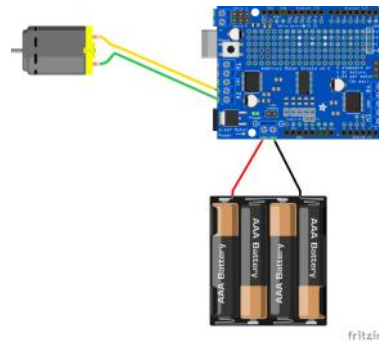
This is how the setup will be constructed. We have the motor shield on top of the Arduino connected to the batteries, and connected to a motor.

Amazing.

MOTORS:

Motors are a relatively easy tool to work with. You'll have to set their speed in the ide, and they'll connect to the board through the L293D motor controller.

REMOTE CONTROLLING:

To be able to move, our rover must be connected to the computer, so we must be able to use a motor controller. In the phases of design, I'll use a usb from the computer to control the board, then I will develop a control app for the car.

The controller is called an HC-05 or HC-06 Bluetooth Module. It's the easiest way to connect to a car from a phone.