

0 - Directions

Thursday, August 8, 2019 1:08 PM

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

Thursday, August 8, 2019 1:04 PM

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

Friday, September 20, 2019 12:28 PM

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

Thursday, August 8, 2019 1:21 PM

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	

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


Thursday, August 8, 2019 1:03 PM

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_LED
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.

```
1. class Flasher  
2. {  
3.     // Class Member Variables  
4.     // These are initialized at startup  
5.     int ledPin; // the number of the LED pin  
6.     long onTime; // milliseconds of on-time  
7.     long offTime; // milliseconds of off-time  
8.     // These maintain the current state  
9.     int ledState; // ledState used to set the LED  
10.    unsigned long previousMillis; // will store last time LED was updated  
11.    // Constructor - creates a Flasher  
12.    // and initializes the member variables and state  
13.    public:  
14.    Flasher(int pin, long on, long off)  
15.    {  
16.        ledPin = pin;  
17.        pinMode(ledPin, OUTPUT);  
18.        onTime = on;  
19.        offTime = off;  
20.        ledState = LOW;  
21.        previousMillis = 0;  
22.    }  
23.    void update()  
24.    {  
25.        // check to see if it's time to change the state of the LED  
26.        unsigned long currentMillis = millis();  
27.        if (ledState == HIGH) { // currentMillis - previousMillis == onTime  
28.            {  
29.                ledState = LOW; // turn it off  
30.                previousMillis = currentMillis; // Remember the time  
31.                digitalWrite(ledPin, ledState); // update the actual LED  
32.            }  
33.        }  
34.        else if ((ledState == LOW) && (currentMillis - previousMillis == offTime))  
35.        {  
36.            ledState = HIGH; // turn it on  
37.            previousMillis = currentMillis; // Remember the time  
38.            digitalWrite(ledPin, ledState); // update the actual LED  
39.        }  
40.    }  
41.    // Flasher led1(12, 1000, 400);  
42.    // Flasher led2(13, 200, 200);  
43.    void setup()  
44.    {  
45.        //  
46.    }  
47.    void loop()  
48.    {  
49.        led1.update();  
50.        led2.update();  
51.    }  
52. }
```

This is a sample program to work with LEDs.

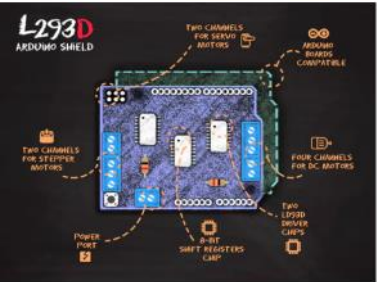
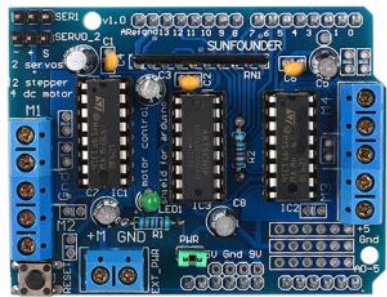
You must still have the setup() and loop() methods.

HARDWARE SELECTION:

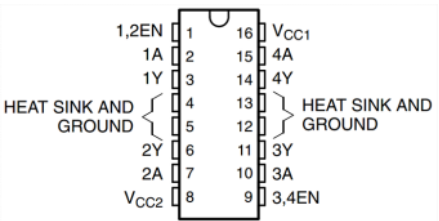
Motor Controller:

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 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

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.