



Robotic Merit Badge Session #1

- ▼ Merit Badge Counselor: Maurice Ling
- ▼ July 6, 2015
- ▼ <http://bsatroop675.org>



Agenda

- ▼ Class Overview
 - ▼ Schedule
 - ▼ Materials for the Class
 - ▼ Robotics Merit Badge Requirements
 - ▼ Competition
- ▼ Safety Working in Robotics
- ▼ Introduction to Your Robot Brain (Dagu Micromagician)
- ▼ Programming
 - ▼ Development Tips
 - ▼ Exercises
- ▼ Assignment to work on before Session #2



Schedule

Class Dates	Topics
7/6	Introduction, Material Distribution, Programming
7/20	Programming, Electronics, Competition Specifications, Design
8/3	Design Review, Electronics
8/17	Pre-competition Preparation, Advanced Topics
8/31	Robotics Showcase/Competition



Computer Setup

▼ Review Preparations for the First Session

- ▼ Arduino IDE Installation
- ▼ USB Driver Installation
- ▼ Github account
 - ▼ Fork <https://github.com/mcli/RoboticsMB>
 - ▼ Clone your RoboticsMB project
 - ▼ Launch Git Shell or Terminal
 - ▼ Setup upstream project:

```
git remote add upstream https://github.com/mcli/RoboticsMB.git
```

▼ Synchronize upstream project (when needed)

```
git fetch upstream  
git checkout master  
git merge upstream/master
```



GitHub

- ▼ GitHub is a source code control service, based on the “git” utility
- ▼ I will use it to look at your programs to monitor your progress
- ▼ Commit and synchronize your code to make it available in your GitHub repository.
- ▼ Commit your code when you reach a stable point in development to checkpoint your changes.
- ▼ You can also synchronize with my project to get any class examples or utilities.
- ▼ Fork <https://github.com/mcli/RoboticsMB> and make it your upstream project (See Preparing for the First Session).



Material Inventory

- ▼ Carrying Case
- ▼ Tadpole Robot Kit
 - ▼ The Tadpole is only the starting point for your robot design.
 - ▼ You will be modifying it to achieve your own design goals.
- ▼ 4 AAA NiMH batteries
- ▼ Engineering notebook
- ▼ Merit Badge Booklet
- ▼ Merit Badge Workbook



Scout-Provided Materials

- ▼ Blue Card signed by Scoutmaster
- ▼ Computer with Arduino development environment installed
- ▼ USB-A to mini-USB Cable
- ▼ AAA NiMH battery charger
- ▼ Small phillips screw driver
- ▼ Pen/Pencil for taking notes



Other Useful Items

- ▼ Sony/Universal Remote Control
- ▼ Electric Multimeter to measure Voltage and Resistance
- ▼ Solder and Soldering Iron



Merit Badge Requirements

All Requirements are listed in the Workbook and Merit Badge Booklet and cover:

- ▼ Safety
- ▼ Robotics in industry
- ▼ General Knowledge
- ▼ Demonstration
- ▼ Competitions
- ▼ Careers

Complete the workbook on your own for Merit Badge
Counselor review/check-off



Robotics Showcase Competition

- ▼ Gives you objectives for your robot design.
- ▼ You may optionally work together with one other scout as a team.
- ▼ Have fun.
- ▼ More details will be announced at the 2nd Session



Safety Working with Robotics



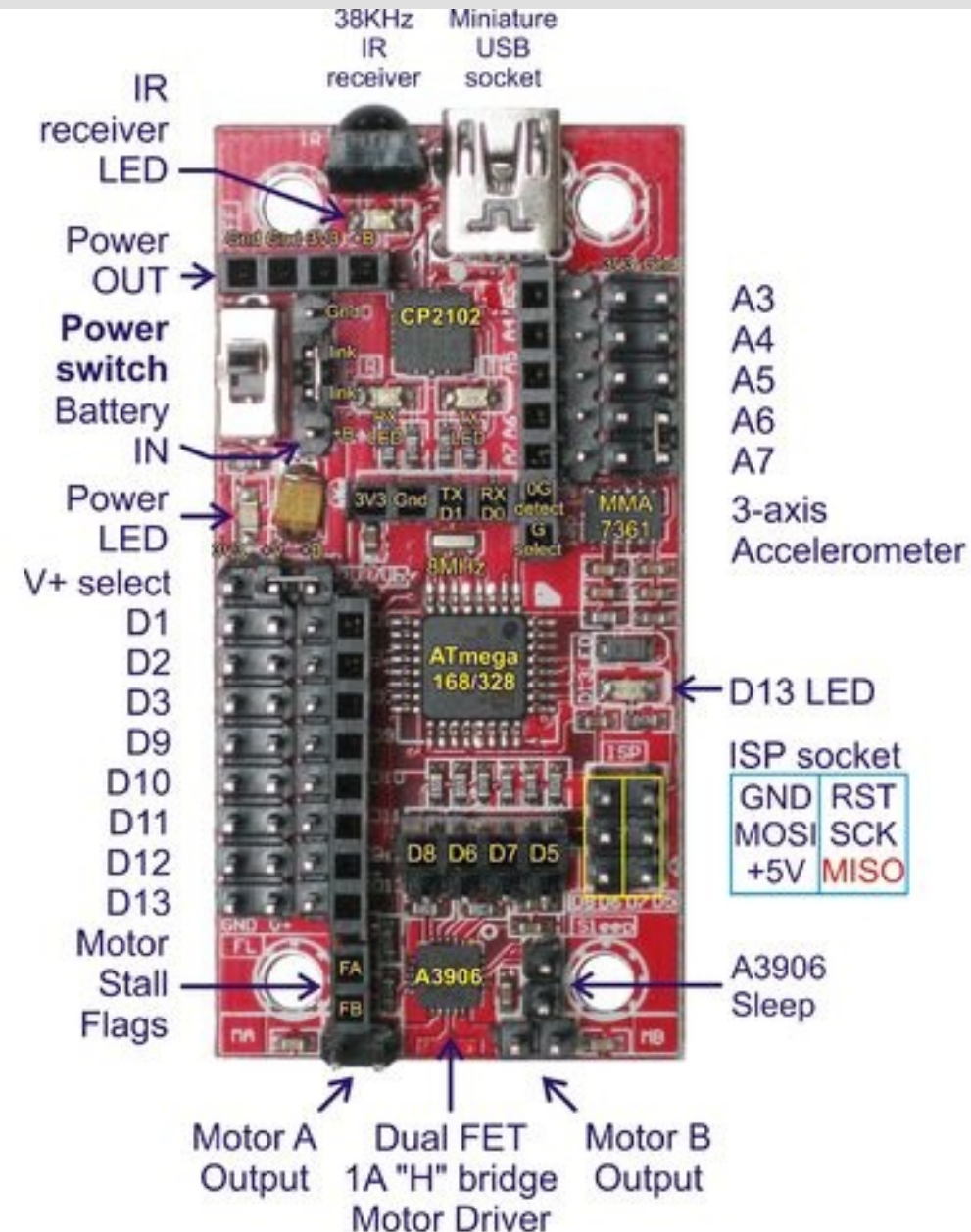
Safety Working with Robotics

- ▼ Have a clean work area
- ▼ Electrical Circuits
 - ▼ Always disconnect power when modifying circuits.
- ▼ Electro-static discharge
- ▼ Precautions when cutting material



Introduction to Your Robot Brain

- ▼ The **MicroMagician** Controller
 - ▼ Atmega 328 microprocessor
 - ▼ Digital and Analog pins (I/O)
 - ▼ Power in/out -3.3V vs 5V
 - ▼ Motor driver and connections
 - ▼ Accelerometer
 - ▼ IR receiver



Analog Vs Digital Signals

- ▶ Analog signals are continuous with many values between minimum and maximum. e.g. a sine wave. Typically used for sensor processing.
- ▶ Digital signals are discrete. e.g. 0 or 1. Typically used to turn something on or off.



Analog Signal



Digital Signal



Development Tips

- ▼ Using the IDE to compile and upload programs (“sketches”)
- ▼ Using the Reference: Help → Reference
 - ▼ setup() - Runs once
 - ▼ loop() - Gets called continuously
 - ▼ ; (semicolon)
 - ▼ Data Types
 - ▼ = (assignment) vs == (equal to)
- ▼ All code must be within a function
- ▼ Utilizing libraries (microM Library)



Programming Exercises - Blink

- ▼ Load the Blink program from Examples → 01.Basics
- ▼ Save it in your sketchbook
- ▼ Compile it and upload it to the MicroMagician Board

```
void setup() {  
  // initialize digital pin 13 as an output.  
  pinMode(13, OUTPUT);  
}
```

// the loop function runs over and over again forever

```
void loop() {  
  digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(1000);           // wait for a second  
  digitalWrite(13, LOW);  // turn the LED off by making the voltage LOW  
  delay(1000);           // wait for a second  
}
```



Programming Exercises – Modifying Blink

- ▼ Save the pin number (13) in an “int” variable and replace all instances of “13” with the variable. e.g. `int ledPin=13;`
- ▼ Modify the program to generate the SOS signal – 3 short, 3 long, 3 short, pause, then repeat.



Programming Exercises – Accelerometer

- ▼ Load the MicroM example “impact”: Files → libraries → microM → impact
- ▼ Open the Serial Monitor: Tools → Serial Monitor
- ▼ Run program, then tap the board lightly from different directions to see the delta values change
- ▼ Save the program into your sketches
- ▼ Modify the program to print the x, y, and z axis readings (not the delta readings)
- ▼ Turn the board in different directions and observe the readings.



Programming Exercises – Accelerometer

```
#include <microM.h>
```

```
void setup()  
{
```

```
  microM.Setup();  
  microM.sensitivity=50;
```

```
  microM.devibrate=50;
```

```
  Serial.begin(9600);  
  Serial.println("Ready");
```

```
}
```

```
// this must be called first in your setup function  
// if your robot vibrates or moves over a rough surface  
// then increase this value to prevent false triggering  
// depends on the construction of your robot. Increase  
// the value if you get additional readings after initial impact
```

```
// initialize serial monitor
```



Programming Exercises – Accelerometer

```
void loop()
{
  microM.Impact();           // function must be called at least once every 2mS to
                             //work accurately
  If (microM.magnitude>0)    // display results of impact
  {
    Serial.println("");
    Serial.print("\tMagnitude:");
    Serial.print(microM.magnitude);
    Serial.print("\tDelta X:");
    Serial.print(microM.deltx);
    Serial.print("\tDelta Y:");
    Serial.print(microM.delty);
    Serial.print("\tDelta Z:");
    Serial.println(microM.deltz);
    Serial.println("");
    microM.magnitude=0;      // prevents display repeating the same values
  }
}
```



Assignment

- ▼ Finish Class Exercises
- ▼ Commit and sync your programming exercises to your GitHub RoboticsMB project.
- ▼ Put together your tadpole robot following the instructions at [instructables.com](https://www.instructables.com). Note that our Tadpole kit does not contain the bumper switches.
- ▼ Follow the tutorials at [RocketBrandStudios](https://www.RocketBrandStudios.com) to program and calibrate the motors.
- ▼ If you have Sony or universal remote control, use the microM → IR Receiver project to determine what codes correspond to which buttons on your remote. Write the button names/functions and the corresponding codes in a table in your engineering notebook. See if you can control your robot using the IR remote control.

