Robotic Merit Badge Session #2

- Merit Badge Counselor: Maurice Ling
- July 20, 2015
- http://bsatroop675.org

Agenda

- Homework Review
- Competition Details
- Electronics
  - Hooking up the Obstacle Sensor
- Programming
  - Reading the Obstacle Sensor inputs
  - Exercises
- Session #2 Homework

Homework Review

- Programming with Functions
- Functions with arguments
- Calibrating Motors
- IR Control
- Using GitHub

Programming with Functions

```
void setup()
{
 // same setup as before
}
void shortBlinks()
{
  for (int i = 0; i < 3; i++)
  {
    // call digitalWrite/delay here set pin HIGH, then LOW
  }
}

// define void longBlinks() here, similar to shortBlinks()

void loop()
{
   shortBlinks();
   longBlinks();
   shortBlinks();
   delay(1000);
}
```

Functions with Arguments

```
const int ledPin=13;
 // same setup() as before
void blinks(int length)
{
      for (int i = 0; i < 3; i++)              // Loop three times
      {
            digitalWrite(ledPin, HIGH); // turn on
            delay(length);
            digitalWrite(ledPin, LOW); // turn off
            delay(length);
      }
}

void loop()
{
      const int longLength=1000;
      const int shortLength=500;
      blinks(shortLength);
      blinks(longLength);
      blinks(shortLength);
      delay(longLength); // pause before repeating sequence
}
```

Calibrating The Motors

- Connections to Motor A Output and Motor B Output
- microM.motors(*leftSpeed,rightSpeed,leftBrake,rightBrake);*
- Figuring out the Right and Left Motors
- Making your robot go straight.

Looking Under the Hood at the microM Library

- microM.h – Header file defines the microM C++ class and methods (functions) you can use.
- microM.cpp – The implementation of the C++ class

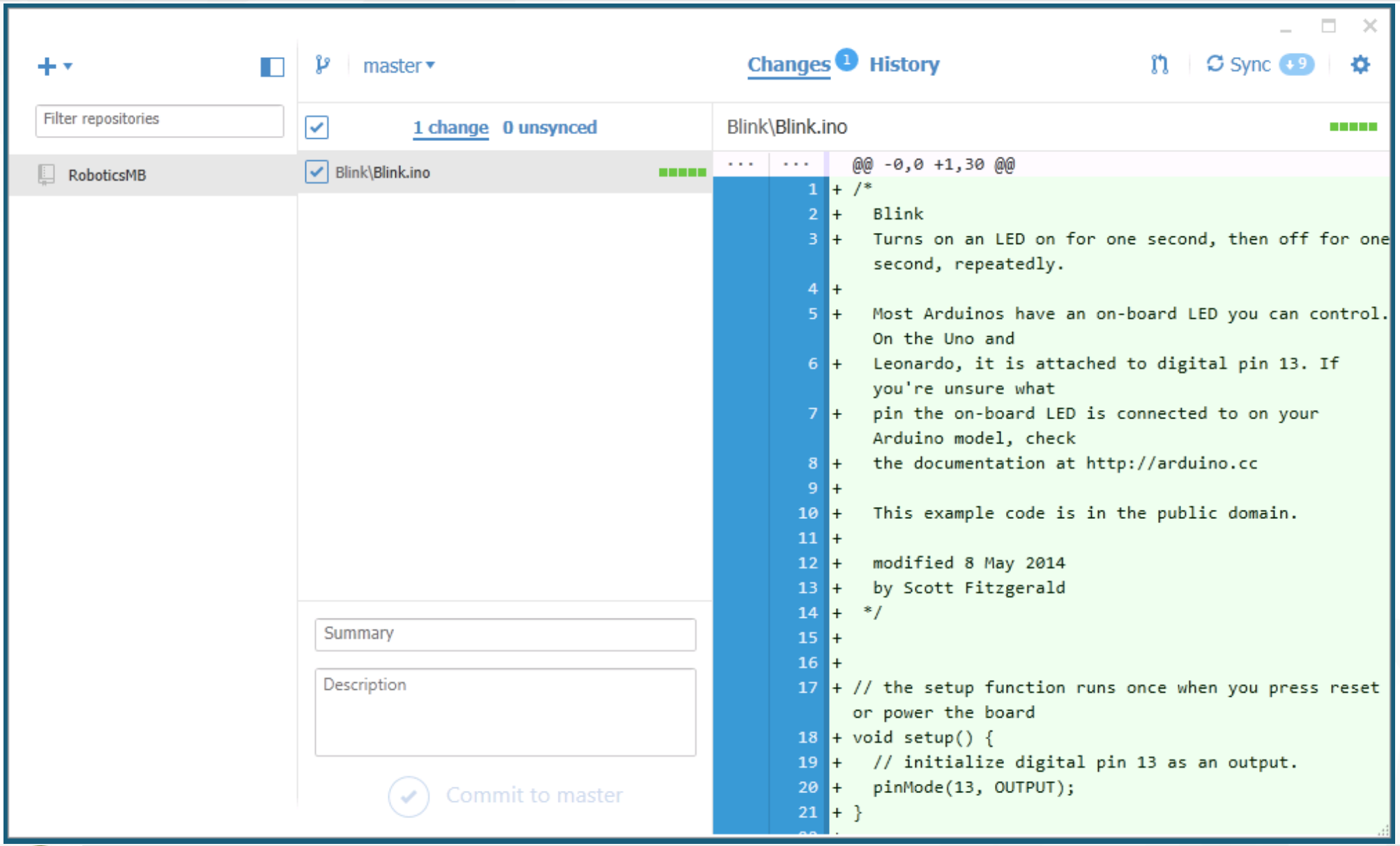
```
void MicroM::Motors(int leftSpeed, int rightSpeed, byte leftBrake, byte rightBrake)
{
        pinMode(7,OUTPUT);
        PinMode(8,OUTPUT);
        if(leftBrake!=0)
        {
                digitalWrite(5,1);
                digitalWrite(7,1);
        }
        else
        {
                If (leftSpeed==0)
                {
                        digitalWrite(5,0);
                        digitalWrite(7,0);
                }
                I f(leftSpeed>0)
                {
                        analogWrite(5,leftSpeed);
                        digitalWrite(7,0);
                }
                If (leftSpeed<0)
                {
                        leftSpeed+=255;
                        analogWrite(5,leftSpeed);
                        digitalWrite(7,1);
                }
        }
...
```

Remote Key to IR Command Table

Remote Key	IR Command Code
1-9	1-9
0	10
Left	124
Right	125
Up	122
Down	123
Enter	12
Play	51
Pause	58
Stop	57

- Go Straight Example

GitHub Client



Announcing the 2015 Troop 675 Robotics Competition!

Robotic Plutonium Carry

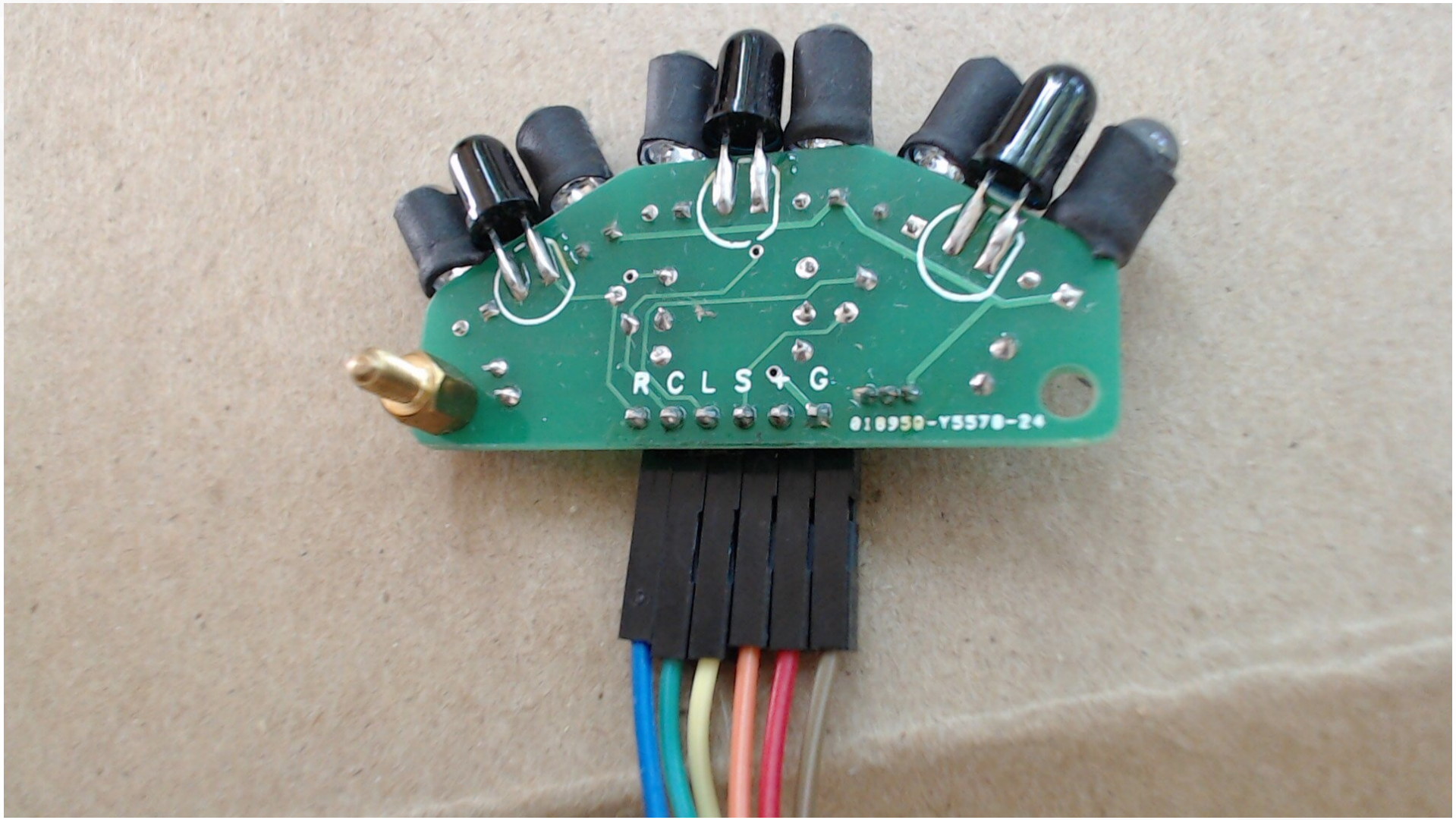
- Objective: Transport radioactive plutonium payload from one reactor to another.
- Payload will take the form of a ping pong ball
- Begin and end with a human-controlled segment to load and unload the capsule.
- Carefully navigate a maze autonomously
- Option to team up with another scout
- Complete rules in github.

Competition Scoring

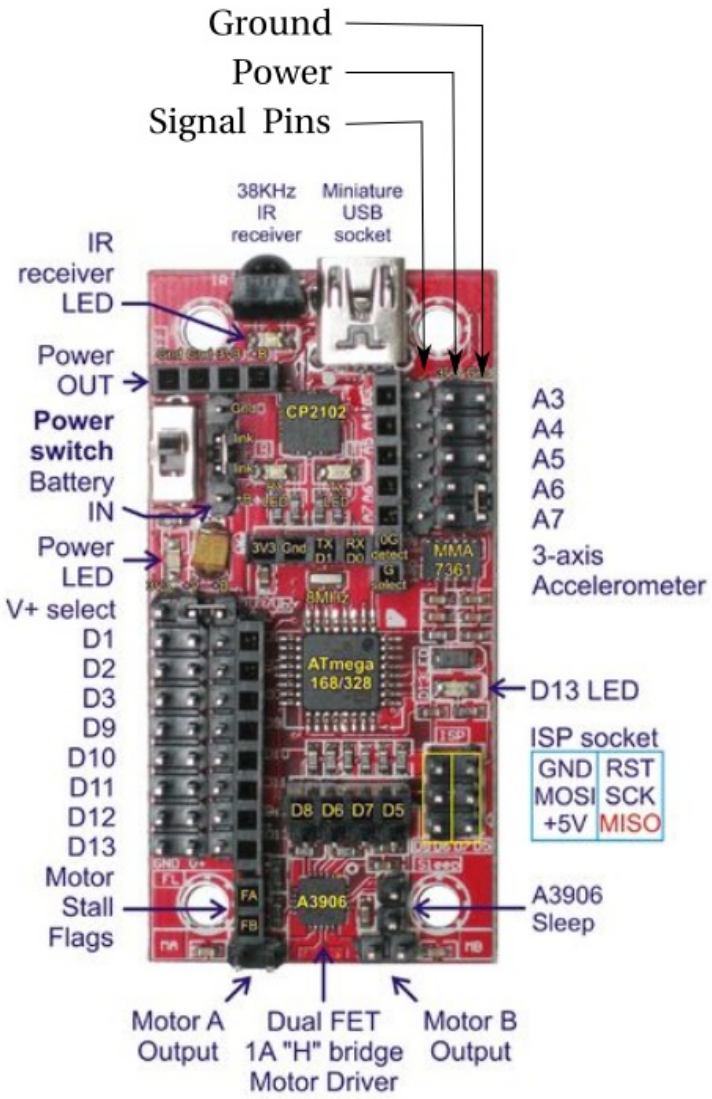
- +20 Points scored for each segment completed
- Bonus points for time completed within 3 minutes
- Bonus Design points
- Penalty points for handling payload and manual intervention
- Parent Judge panel

Obstacle Sensor

Obstacle Sensor



Obstacle Sensor Connections



- Connect to Analog Pins
- One Ground
- One Power (+3.3V)
- One Digital signal output
  - D1 - on/off
- Three Analog Inputs
  - A3 - Right,
  - A4 - Center,
  - A5 - Left

Processing Signals from the Obstacle Sensor

- Setup pin 1 for output
- Turn on the sensor by using digitalWrite(1,HIGH)
- Use readAnalog(*pinNumber)* to read the signal from the three different sensors.
- Observe the values when you move your hand in front of the different sensors. Move your hand close, then move it far away.
- What are the values when nothing is in front of the sensor?

Reading the Obstacle Sensors

```
const int leftPin=3;
const int centerPin=4;
const int rightPin=5;
const int senseOnOffPin=1;

void setup()
{
  pinMode(senseOnOffPin,OUTPUT); // set up sensor on/off pin
  Serial.begin(19200); // set the serial port speed.
  digitalWrite(senseOnOffPin, HIGH);  // Turn on the sensor
}

void loop()
{
   int sensorInput=0;
   sensorInput=analogRead(leftPin); //left
   Serial.print(sensorInput);
   sensorInput=analogRead(centerPin); //center
   Serial.print(sensorInput);
   sensorInput=analogRead(rightPin); //right
   Serial.print(sensorInput);
   Serial.println();
}
```

Exercises

- Normalize the sensor readings so that if there is nothing in front of the sensor, the reading is close to zero. Use separate normalization offset values for left, center, and right.
- Write a function that prints out “Detect Left” when there is something on the left, “Detect Right” when there is something on the right, and “Detect Straight” if there is something straight ahead.
- Write a function to control the motors to avoid obstacles if they come within a certain range.
- Add the capability to use the IR remote to switch between autonomous and manual control.

Helpful Hint – Using Arrays

```
const int numSensors=3;
const int sensePin[3]={5,4,3};  // array containing the pin numbers
const int senseOnOffPin=1;
int sensorInput[3]={0}; // initialize sensor inputs to 0

// setup() function omitted for brevity

void loop()
{
  for (int i = 0; i < numSensors; i++)
  {
    sensorInput[i]=analogRead(sensePin[i]);
    Serial.print(sensorInput[i]);
    Serial.print(" ");
  }
  Serial.println();
}
```

Session #2 Homework

- Complete class exercises
- In your notebook, capture the design for the competition
  - How to load/unload the payload
  - How to carry the payload
  - Sketch the logic required to navigate around a maze
- Program your robot to implement your maze navigation logic
- Robotics MB Workbook:
  - Section 3
  - Section 4 a, b