Accurate Movement (We Like to Move It!)

Time required: 90 minutes

Please read all the directions carefully before beginning the assignment.

1. Comment your code as shown in the tutorials and other code examples.
2. Follow all directions carefully and accurately.
3. Think of the directions as minimum requirements.

Understanding

Demonstrate understanding of:

libraries, functions

We know how much time it takes to move a certain distance at a certain power. We can input the distance for accurate movement. We can also calculate the amount of time it take to turn a specific angle.

We will create a resuable library file called **Movement.h** We will use this file to store our movement code and copy it from sketch to sketch. This allows for easily resuable code.

The following are the calculations we use to determine how far we are traveling.

avgSpeed (inches per second) = (Distance(inches) / Time)

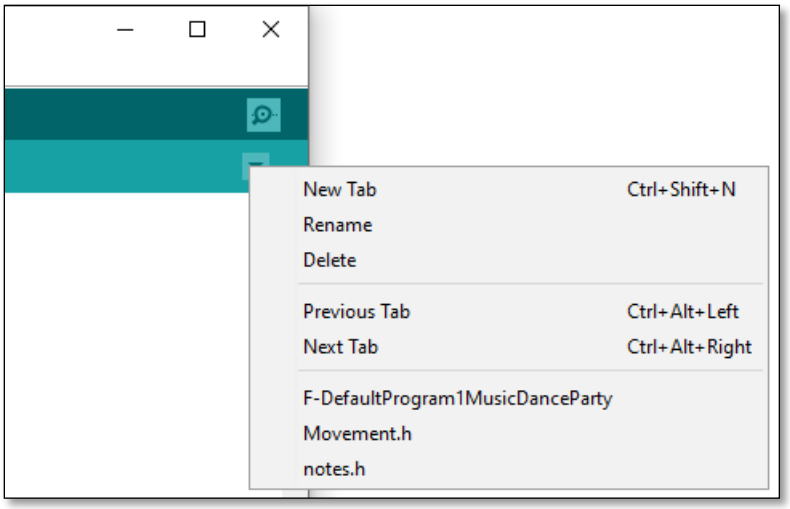
Example: (4’ / 7400) = 6.5 inches per second

Requirements

1. Create a function that tests each of the movements.

Tutorial Assignment

1. Open **CalibrateMovement**. Save the sketch as **AccurateMovement**.
2. On the right side of the Arduino IDE, click the down triangle → Click **New Tab** → **Filename** → **Movement.h** Click OK.



3. Cut and paste the code from the top of the main ino file to **Movement.h**. Look at the code at the end of this document to tell which code to copy and paste.
4. You can delete the code at the bottom of the ino file.
5. Complete and test the program as pictured with the requirements listed.

```ino
/**
   @file    AccurateMovement.ino
   @author  William A Loring
   @version V1.0.0
   @date revised 03/10/2018  created: 12/10/16
   @Description: Accurate mBot movement with methods
*/
#include <MeMCore.h>   // Include mBot library
#include "Movement.h"  // Include custom Movement.h function library
MeIR ir;               // Create IR remote object

void setup() {
  ir.begin();  // Start listening to the remote
}

void loop() {
  remote();  // Check remote for button press
}

// Determine if a remote button is pressed
void remote() {
  if (ir.keyPressed(IR_BUTTON_UP)) {           // If a remote button is pressed
    moves();                                   // Call moves function
  } else if (ir.keyPressed(IR_BUTTON_DOWN)) {  // If a remote button is pressed
    yourMoves();                               // Call new function
  }
}
```

```
// Combination of movement functions from the Movement.h file
void moves() {
  forwardInches(12);
  reverseInches(12);
  stop();
  delay(1000);  // This is an Arduino function
  forwardInches(12);
  leftTurnDegrees(90);
  forwardInches(12);
  rightTurnDegrees(90);
  forwardInches(12);
}

// Combination of your moves from the Movement.h file
void yourMoves() {
  // Insert your move functions here

}
```

Movement.h

```
/**
    @file    Movement.h
    @author  William A Loring
    @version V1.0.0
    @date Revised 04/03/18  Created: 12/07/17
    @Description: Portable mBot movement with methods library file
*/
#include <MeMCore.h>  // Include mBot library

// Create motor control objects
MeDCMotor MotorL(M1);   // MotorL is Left Motor
MeDCMotor MotorR(M2);   // MotorL is Right Motor
const int POWER = 127;  // Base power setting at 50% Maximum is 255

// Use forward48() to calibrate distance
// Increase COMP .02 at a time if your robot drives to the left
// Decrease COMP .02 at a time if your robot drives to the right
const float COMP = 1.0;  // Compensation to make the robot drive straight

// Apply compensation to left motor
// Use round function to convert float result to integer
int lPower = round(POWER * COMP);  // Apply compensation to left motor
int rPower = POWER;

// Increase time if the robot comes up short, decrease if it goes too far
const int DRIVE_TIME = 5400;  // Time in milliseconds it takes to go 48"

// Use the average of leftSquare() and rightSquare() to calibrate turns
// Increase by 20 ms at a time if the robots 90 degree is short
// Decrease by 20 ms at a time if the robots 90 degree is too long
const int TURN_TIME = 530;  // Time in milliseconds it takes to turn 90 degrees

const int DISTANCE = 48;
// Calculate inches per second
// (float) casts DISTANCE int constant to a float
// This forces float math instead of integer math
float inchPerSec = (float)DISTANCE / DRIVE_TIME;
```

```
39  //-------------------------------------------------------------------------------------------
40  // Stop function: This function is called in other functions, it has to be first
41  void stop() {
42    MotorL.stop();  // Stop MotorL
43    MotorR.stop();  // Stop MotorR
44  }
45
46  //-------------------------------------------------------------------------------------------
47  // Forward function with distance in inches argument
48  void forwardInches(int distance) {
49    float drvTime;                    // Time it takes to drive a certain distance
50    drvTime = distance / inchPerSec;  // Calculate drive time in milliseconds
51    MotorL.run(-lPower);              // MotorL (Left)  forward is -negative
52    MotorR.run(+rPower);              // MotorR (Right) forward is +positive
53    delay(drvTime);                   // Drive a certain number of inches based on avgSpeed
54    stop();                           // Stop Motors
55  }
56
57  //-------------------------------------------------------------------------------------------
58  // Reverse function with distance in inches argument
59  void reverseInches(int distance) {
60    float drvTime;                    // Time it takes to drive a certain distance
61    drvTime = distance / inchPerSec;  // Calculate drive time in milliseconds
62    MotorL.run(+lPower);              // MotorL (Left)  reverse is +positive
63    MotorR.run(-rPower);              // MotorR (Right) reverse is -negative
64    delay(drvTime);                   // Drive a certain number of inches based on avgSpeed
65    stop();                           // Stop Motors
66  }
```

```
68  //--------------------------------------------------------------------------------
69  // Left turn function with degrees of turn argument
70  void leftTurnDegrees(int degrees) {
71    float drvTime;                          // Time it takes to drive a certain distance
72    drvTime = (degrees / 90.0) * TURN_TIME; // Calculate turn time for degrees
73    MotorL.run(+lPower);                    // MotorL (Left) reverse is +positive
74    MotorR.run(+rPower);                    // MotorR (Right) forward is +positive
75    delay(drvTime);                         // Turn a certain number of degrees based on time
76    stop();                                 // Stop Motors
77  }
78
79  //--------------------------------------------------------------------------------
80  // Right turn function with degrees of turn argument
81  void rightTurnDegrees(int degrees) {
82    float drvTime;                          // Time it takes to drive a certain distance
83    drvTime = (degrees / 90.0) * TURN_TIME; // Calculate turn time for degrees
84    MotorL.run(-lPower);                    // MotorL (Left) forward is -negative
85    MotorR.run(-rPower);                    // MotorR (Right) reverse is -negative
86    delay(drvTime);                         // Turn a certain number of degrees based on time
87    stop();                                 // Stop Motors
88  }
```

Assignment

Start with your tutorial project and add the following.

1. Create another function in the main sketch like the move function with different moves from the **Movement.h** file.
2. Use a different remote key to trigger the new function.

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.