Default Program 1: Remote Control Arduino

Time required: 120 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:

functions

Tutorial Assignment

Let's begin building the mBot default program in Arduino starting with remote control.

1. Open the **DrivingSchool2** program and save it as **DefaultProgram1**.
2. The **Movement.h** file should still be in the program folder.
3. Include the **notes.h** file in the sketch folder. Use this for audio feedback for the sketch.
4. Complete and test the program with the requirements listed.

Requirements

- Base your mode switching on your mBlock Default program.
- Add the necessary variables. For example, **int modeFlag = 0;** to track the mode in the main sketch.
- Create the **setMode**, **remoteControl**, **setSpeed, and speedSet** functions in the main sketch.
- Add the changes shown in the **Movement.h** file.
- Add the **playNote** function from previous Arduino programs to the main sketch. This allows you to easily add audio feedback.

```ino
/**
   @file    DefaultProgram1.ino
   @author  William A Loring
   @version V1.0.0
   @date revised 11/13/2020  created: 12/10/16
   @Description: Part 1 of mBot Default Program, remote control
*/
#include <MeMCore.h>      // Include mBot library
#include "Movement.h"     // Include custom Movement function library
#include "notes.h"        // Include notes library for ease of creating sounds
MeIR ir;                  // Create ir remote object
MeBuzzer buzzer;          // Create buzzer object
MeRGBLed led(0, 30);      // Create onboard LED object
int modeFlag = 0;         // Flag to track the state of robot Mode
const int DEBOUNCE = 50;  // Time it takes to debounce the ir remote keys

void setup() {
  ir.begin();      // Start listening to the remote
  led.setpin(13);  // Set the Arduino pin for the led's
  initialize();    // Play initialization sounds and show LED's
}

void loop() {
  setMode();  // Check ir remote for mode setting
  // If button A is pressed, remote control mode
  if (modeFlag == 0) {
    // Check for modeFlag set to 0 for Remote control operation if Button A is pressed
    remoteControl();
  }
}

//-------------------------------------------------------------------------------
// Determine the robot's mode of operation, A or modeFlag 0 - Remote Control is default
//-------------------------------------------------------------------------------
void setMode() {
  // Determine which remote button was pressed
  if (ir.keyPressed(IR_BUTTON_A)) {
    delay(DEBOUNCE);
    modeFlag = 0;          // Set Mode A, Remote Control
    playNote(noteC4, HN);  // Play note to indicate mode change
  }
}
```

```
//-------------------------------------------------------------------------------------------
// Remote Control functions
//-------------------------------------------------------------------------------------------
void remoteControl() {
  // Set the speed of the mBot
  setSpeed();
  if (ir.keyPressed(IR_BUTTON_UP)) {
    delay(REMOTE_DEBOUNCE);
    forward();               // Move forward
    led.setColor(0, 30, 0);  // Set both LED to Green
    led.show();
  } else if (ir.keyPressed(IR_BUTTON_DOWN)) {
    delay(REMOTE_DEBOUNCE);
    reverse();               // Move backwards
    led.setColor(30, 0, 0);  // Set both LED to Red
    led.show();
  } else if (ir.keyPressed(IR_BUTTON_LEFT)) {
    delay(REMOTE_DEBOUNCE);
    left();
    led.setColorAt(1, 0, 30, 0);  // Set Left LED to Green
    led.setColorAt(0, 0, 0, 0);   // Set Right LED off
    led.show();
  } else if (ir.keyPressed(IR_BUTTON_RIGHT)) {
    delay(REMOTE_DEBOUNCE);
    right();
    led.setColorAt(1, 0, 0, 0);   // Set Left LED off
    led.setColorAt(0, 0, 30, 0);  // Set Right LED to Green
    led.show();
  } else {
    delay(DEBOUNCE);  // Longer delay for remote control to work
    stop();
    led.setColor(0, 0, 0);  // Set both LED's off
    led.show();
  }
}
```

```
82  //--------------------------------------------------------------------------------
83  // Set the robot's speed using the number on the remote control
84  //--------------------------------------------------------------------------------
85  void setSpeed() {
86    if (ir.keyPressed(IR_BUTTON_0)) {
87      // Call the speedSet function with percent of power and the note played
88      speedSet(100, noteC5);
89    } else if (ir.keyPressed(IR_BUTTON_1)) {
90      speedSet(25, noteA3);
91    } else if (ir.keyPressed(IR_BUTTON_2)) {
92      speedSet(30, noteB3);
93    } else if (ir.keyPressed(IR_BUTTON_3)) {
94      speedSet(35, noteC4);
95    } else if (ir.keyPressed(IR_BUTTON_4)) {
96      speedSet(40, noteD4);
97    } else if (ir.keyPressed(IR_BUTTON_5)) {
98      speedSet(50, noteE4);
99    } else if (ir.keyPressed(IR_BUTTON_6)) {
100     speedSet(60, noteF4);
101   } else if (ir.keyPressed(IR_BUTTON_7)) {
102     speedSet(70, noteG4);
103   } else if (ir.keyPressed(IR_BUTTON_8)) {
104     speedSet(80, noteA4);
105   } else if (ir.keyPressed(IR_BUTTON_9)) {
106     speedSet(90, noteB4);
107   }
108 }
```

```
//-------------------------------------------------------------------------------------
// Set speed function with notes
void speedSet(int speedInc, int notes) {
  int power = 0;
  delay(DEBOUNCE);
  power = SPEED_FACTOR * speedInc;
  setPower(power);
  playNote(notes, HN);
}

//-------------------------------------------------------------------------------------
void playNote(int note, int duration)
// This custom function takes two parameters, note and duration to make playing songs easier
// Each of the notes have been #defined in the notes.h file. The notes are broken down by
// octave and sharp (s) / flat (b).
{
  buzzer.tone(note, duration);
}

void initialize() {
  // Play initialization notes and led's
  delay(DEBOUNCE);
  led.setColor(30, 0, 0);  // Set both LED to Red
  led.show();
  playNote(noteC4, HN);
  led.setColor(0, 0, 30);  // Set both LED to Blue
  led.show();
  delay(50);
  playNote(noteD4, HN);
  delay(50);
  playNote(noteD4, HN);
  led.setColor(30, 0, 0);  // Set both LED to Green
  led.show();
  playNote(noteE4, QN);
  delay(50);
  playNote(noteE4, QN);
  delay(50);
  playNote(noteE4, QN);
  led.setColor(0, 0, 0);  // Turn both LED's off
  led.show();
}
```

Movement.h

```
/**
    @file    Movement.h
    @author  William A Loring
    @version V1.0.0
    @date Revised 10/30/20  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
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);
int rPower = POWER;
const int TURN_TIME = 530;    // Time in milliseconds to turn 90 degrees right
const int DRIVE_TIME = 5400;  // Time in milliseconds to go 48"
const int DISTANCE = 48;      // 48"
// Calculate inches per second
// (float) converts the integer DISTANCE to a float,
// otherwise there would be integer math
float inchPerSec = (float)DISTANCE / DRIVE_TIME;
// Set to this number for maximum speed to go straight with COMP
const float SPEED_FACTOR = 2.42;  // Constant to change speed with remote

//-------------------------------------------------------------------------------
// Reset power variables for remote speed control
void setPower(int pwr) {
  // Use round function from math.h to convert float result to integer
  lPower = round(pwr * COMP);  // Apply compensation to left motor
  rPower = pwr;                // Set right motor power
}

//-------------------------------------------------------------------------------
// Stop function: because this function is called in other functions,
// it has to be first
void stop() {
  MotorL.stop();  // Stop MotorL
  MotorR.stop();  // Stop MotorR
}
```

```
//-------------------------------------------------------------------------------------
// Forward function for remote and line following
void forward() {
  MotorL.run(-lPower);  // MotorL (Left)  forward is -negative
  MotorR.run(+rPower);  // MotorR (Right) forward is +positive
}

//-------------------------------------------------------------------------------------
// Reverse function for remote and line following
void reverse() {
  MotorL.run(+lPower);  // MotorL (Left)  reverse is +positive
  MotorR.run(-rPower);  // MotorR (Right) reverse is -negative
}

//-------------------------------------------------------------------------------------
// Left turn function for remote and line following
void left() {
  MotorL.run(+lPower);  // MotorL (Left) reverse is +positive
  MotorR.run(+rPower);  // MotorR (Right) forward is +positive
}

//-------------------------------------------------------------------------------------
// Right turn function for remote and line following
void right() {
  MotorL.run(-lPower);  // MotorL (Left) forward is -negative
  MotorR.run(-rPower);  // MotorR (Right) reverse is -negative
}
```

```
72  //-------------------------------------------------------------------------------
73  // Forward function with distance in inches argument
74  void forwardInches(int distance) {
75    float drvTime;                    // Time it takes to drive a certain distance
76    drvTime = distance / inchPerSec;  // Calculate drive time in milliseconds
77    MotorL.run(-lPower);              // MotorL (Left)  forward is -negative
78    MotorR.run(+rPower);              // MotorR (Right) forward is +positive
79    delay(drvTime);                   // Drive certain number of inches based on avgSpeed
80    stop();                           // Stop Motors
81  }
82
83  //-------------------------------------------------------------------------------
84  // Reverse function with distance in inches argument
85  void reverseInches(int distance) {
86    float drvTime;                    // Time it takes to drive a certain distance
87    drvTime = distance / inchPerSec;  // Calculate drive time in milliseconds
88    MotorL.run(+lPower);              // MotorL (Left)  reverse is +positive
89    MotorR.run(-rPower);              // MotorR (Right) reverse is -negative
90    delay(drvTime);                   // Drive certain number of inches based on avgSpeed
91    stop();                           // Stop Motors
92  }
93
94  //-------------------------------------------------------------------------------
95  // Left turn function with degrees of turn argument
96  void leftTurnDegrees(int degrees) {
97    float drvTime;                          // Time it takes to drive certain distance
98    drvTime = (degrees / 90.0) * TURN_TIME; // Calculate turn time for degrees
99    MotorL.run(+lPower);                    // MotorL (Left) reverse is +positive
100   MotorR.run(+rPower);                    // MotorR (Right) forward is +positive
101   delay(drvTime);                         // Turn number of degrees based on time
102   stop();                                 // Stop Motors
103 }
104
105 //-------------------------------------------------------------------------------
106 // Right turn function with degrees of turn argument
107 void rightTurnDegrees(int degrees) {
108   float drvTime;                          // Time it takes to drive a certain distance
109   drvTime = (degrees / 90.0) * TURN_TIME; // Calculate turn time for degrees
110   MotorL.run(-lPower);                    // MotorL (Left) forward is -negative
111   MotorR.run(-lPower);                    // MotorR (Right) reverse is -negative
112   delay(drvTime);                         // Turn number of degrees based on time
113   stop();                                 // Stop Motors
114 }
```

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.