```
#include <everytime.h>    // every(100){...} executes code every 100 mSecs, everyu(100){...} every 100 microseconds.
#include <TimedPID.h>
#include <Time.h>
#include <TimeLib.h>
#include <FreqMeasure.h>
#include <FreqCount.h>
#include <CaptureTimer.h>

 /*
 * This program designed for Omicron running on an Arduino 2560 MEGA. This was necessary to get enough I/O for this
robot.
 * Written by S.D. Kaehler on 12/29/17, revised 1/16/18.
 * For Bumpers: 0 (low) indicates no contact; 1 (high) indicates contact.
 * Motor controls are off (0, LOW) or on (1, HIGH)
 */

  byte ledState = LOW;                    // ledState is used to set the LED on or off
  unsigned long ledBlinkTime = 125000;    // Interval: blink (4uS/step, 250 steps/mS)
  unsigned long ledTemp = ledBlinkTime;   // Temp holding variable
  unsigned long PotStepTime = 6250;       // Interval: step pots (4uS/step, 250 steps/mS)
  unsigned long PotTemp = PotStepTime;    // Temp holding variable
  unsigned long BeepTime = 2500000;       // Interval: beep (4uS/step, 250 steps/mS)
  unsigned long BeepTemp = BeepTime;      // Temp holding variable
  unsigned long PrevLoopTime;             // Used to determine how long one loop through the proram takes
  unsigned long LoopTime;

  byte Beeper = 27;                  // Sonalert beeper. Drive HIGH to beep.

  byte LM_FwdEnable = 28;            // Drive motor (outputs) direction control pins
  byte LM_RevEnable = 29;
  byte RM_FwdEnable = 30;
  byte RM_RevEnable = 31;

  byte RF_BumpReset = 32;             // Sensor latch reset (output) pins
  byte LF_BumpReset = 33;
  byte CF_BumpReset = 34;
  byte RearBumpReleset = 35;

  byte RF_BumpSw = 36;                // Bumper switch input pins - HIGH=clear, LOW=impact
  byte LF_BumpSw = 37;
```

```
  byte CF_BumpSw = 38;
  byte RearBumpSw = 39;

  byte RF_BumpLatch = 40;             // Latch state input pins - HIGH=impacted, LOW=clear
  byte LF_BumpLatch = 41;
  byte CF_BumpLatch = 42;
  byte RearBumpLatch = 43;
  byte Bumpers = 0;                   // Holds composite value of four switch latches (= 0 to 15)

  byte LM_SpdSel = 44;                // Left motor digital pot (output) control pins, Chip select, active low
  byte LM_SpdUpDown = 45;             // Set pot direction (HIGH=up or LOW=down)
  byte LM_SpdStep = 46;               // Step pot in selected direction
  unsigned int LM_SpdPot = A0         // Current pot reading (0-1000; Off to Fast)
  unsigned int LM_Encoder = A4;       // Left motor encoder input (0-200Hz=0-2V)

  byte RM_SpdSel = 47;                // Right motor digital pot controls, Chip select, active low
  byte RM_SpdUpDown = 48;             // Control pot direction (HIGH=up or LOW=down)
  byte RM_SpdStep = 49;               // Step pot in selected direction
  unsigned int RM_SpdPot = A1;        // Current pot reading (0-1000; Off to Fast)
  unsigned int RM_Encoder = A5;       // Right motor encoder input (0-200Hz=0-2V)

//-----------------------------------------------------------------------------------------
void setup() {                          // Initialization block

  pinMode(LED_BUILTIN, OUTPUT);       // Use built-in LED as a program heartbeat
  pinMode(LM_Encoder, INPUT);         // Encoder input
  pinMode(RM_Encoder, INPUT);         // Encoder input
  pinMode(Beeper, OUTPUT);            // Use for Sonalert
  digitalWrite(Beeper, HIGH);         // Turn beeper off

  // Set up L & R Motor controls
  pinMode(LM_FwdEnable, OUTPUT);
  pinMode(RM_FwdEnable, OUTPUT);
  digitalWrite(LM_FwdEnable, LOW);    // Disable forward motor controls
  digitalWrite(RM_FwdEnable, LOW);

  pinMode(LM_RevEnable, OUTPUT);
  pinMode(RM_RevEnable, OUTPUT);
  digitalWrite(LM_RevEnable, LOW);    // Disable reverse motor controls
  digitalWrite(RM_RevEnable, LOW);
```

```
pinMode(RF_BumpReset, OUTPUT);      // Set up bumper latch reset lines
pinMode(LF_BumpReset, OUTPUT);
pinMode(CF_BumpReset, OUTPUT);
pinMode(RearBumpReset, OUTPUT);

pinMode(RF_BumpSw, INPUT);          // Set up bumper switch lines
pinMode(LF_BumpSw, INPUT);
pinMode(CF_BumpSw, INPUT);
pinMode(RearBumpSw, INPUT);

ResetBumpLatches()                  // Call function to reset/clear latches

pinMode(RF_BumpLatch, INPUT);
pinMode(LF_BumpLatch, INPUT);
pinMode(CF_BumpLatch, INPUT);
pinMode(RearBumpLatch, INPUT);

pinMode(LM_SpdSel, OUTPUT);         // Configure speed pot1
pinMode(LM_SpdUpDown, OUTPUT);
pinMode(LM_SpdStep, OUTPUT);

SpinDownPot1()

pinMode(RM_SpdSel, OUTPUT);         // Configure speed pot2
pinMode(RM_SpdUpDown, OUTPUT);
pinMode(RM_SpdStep, OUTPUT);

SpinDownPot2()

Serial.begin(9600);                 // Set data rate to 9600 bps
randomSeed(analogRead(A15));        // Seed random number generator
int diceRoll;                       // Random number from 1-6
int coinToss;                       // Heads (0) or Tails (1)

// Now hang out here. Watch bump sensors, blink LED, & beep until a bump sensor is touched
do
  {
  CaptureBumpLatches()              // Read & capture status of each bumper switch
```

```
    ledHeartBeat()                    // Step temp timing variable towards zero

    // Chirp beeper to let someone know the robot is waiting for a sensor hit to start moving
    // Sound beeper for 1/10 sec every 3 seconds
    digitalWrite(Beeper, LOW);
    delay(100);
    digitalWrite(Beeper, HIGH);
    delay(2900)
    }
  while(Bumpers == 0);                // No sensors have been hit so keep looping until one is
  delay(1000);
  ResetBumpLatches()                  // Bumper was hit so clear latches and drop oout of this loop
  PrevLoopTime = millis();
}

//--------------------------------------------------------------------------------------------
void loop() {             // Main loop auto-repeats

  ledHeartBeat()          // Pulse onboard LED

  CaptureBumpLatches()    // Set value of "Bumpers". Check if NOT zero below and act accordingly

  WanderAround()          // Move ahead FORWARD

  ledHeartBeat()          // Pulse onboard LED

  if (Bumpers == 1) {
    RightFrontOnly()      // RF only
    ResetBumpLatches()
    }
  else if ((Bumpers == 2) || (Bumpers == 5) || (Bumpers == 7)) {
    SideCenterFronts()    // CF or LF+RF or LF+CF+RF
    ResetBumpLatches()
    }
  else if (Bumpers == 3) {
    RightCenterFront()    // RF+CF
    ResetBumpLatches()
    }
  else if (Bumpers == 4) {
    LeftFrontOnly()       // LF only
```

```
    ResetBumpLatches()
    }
  else if (Bumpers == 6) {
    LeftCenterFront()     // LF+CF
    ResetBumpLatches()
    }
  else if (Bumpers == 8) {
    RearOnly()            // Rear only
    ResetBumpLatches()
    }
  else if (Bumpers >8) {
    // Any front + rear = sound alarm. Robot is in trouble. Dwell here until latches are manually reset
    digitalWrite(LM_FwdEnable, LOW);      // Disable forward motor controls
    digitalWrite(RM_FwdEnable, LOW);
    digitalWrite(LM_RevEnable, LOW);      // Disable reverse motor controls
    digitalWrite(RM_RevEnable, LOW);
    SpinDownPot1()
    SpinDownPot2()
    do {
      digitalWrite(Beeper, LOW);      // On
      delay(250);
      digitalWrite(Beeper, HIGH);     // Off
      delay(750);
      ledHeartBeat ()         // Pulse onboard LED
    while (Bumpers > 0);
    }
  }
}

//-----------------------------------------------------------------------------------------------------
// FUNCTIONS

void ledHeartBeat() {
  // Cycle LED for a heartbeat to show that the program is still running
  ledTemp -=;                 // Step temp timing variable towards zero
  if (ledTemp == 0) {         // Only execute 1/ledBlinkTime cycles
    ledTemp = ledBlinkTime;   // Reset timing variable
    (ledState = !ledState  // swap states of LED (on or off)
    }
  digitalWrite(LED_BUILTIN, ledState);
```

```
  }
}

void ResetBumpLatches() {   // Reset four bumper latches as long as bump switches are open
  if (RearBumpSw != LOW) {
    digitalWrite(RearBumpReset, HIGH);
    digitalWrite(RearBumpReset, LOW);
    digitalWrite(RearBumpReset, HIGH);
    bitClear(Bumpers,3);
  }
  if (LF_BumpSw != LOW) {
    digitalWrite(LF_BumpReset, HIGH);
    digitalWrite(LF_BumpReset, LOW);
    digitalWrite(LF_BumpReset, HIGH);
    bitClear(Bumpers,2);
  }
  if (CF_BumpSw != LOW) {
    digitalWrite(CF_BumpReset, HIGH);
    digitalWrite(CF_BumpReset, LOW);
    digitalWrite(CF_BumpReset, HIGH);
    bitClear(Bumpers,1);
  }
  if (RF_BumpSw != LOW) {
    digitalWrite(RF_BumpReset, HIGH);
    digitalWrite(RF_BumpReset, LOW);
    digitalWrite(RF_BumpReset, HIGH);
    bitClear(Bumpers,0);
  }
  digitalWrite(Beeper, LOW);    // Chirp beeper at latch reset
  delay(100);
  digitalWrite(Beeper, HIGH);
}

void CaptureBumpLatches() {
  // Read each bumper latch & set respective bits in "Bumpers"
  if (digitalRead(Rear_BumpLatch) == HIGH) {bitWrite(Bumpers,3,HIGH)}
  if (digitalRead(LF_BumpLatch) == HIGH) {bitWrite(Bumpers,2,HIGH)}
  if (digitalRead(CF_BumpLatch) == HIGH) {bitWrite(Bumpers,1,HIGH)}
  if (digitalRead(RF_BumpLatch) == HIGH) {bitWrite(Bumpers,0,HIGH)}
  Serial.print("Bumpers: ");
```

```
  Serial.println(Bumpers);      // Display bumper variable value (0 to 15)
}

void SpinDownPot1() {
  // Fast spin to zero
  digitalWrite(LM_SpdUpDown, LOW);      // Down
  digitalWrite(LM_SpdSel, LOW);         // Set outputs to LM digital pot
  digitalWrite(LM_SpdStep, HIGH);       // Ready to step
  //
  while (analogRead(A0) != 0 {          // Spin pot down to zero
    digitalWrite(LM_SpdStep, HIGH);
    digitalWrite(LM_SpdStep, LOW);
    digitalWrite(LM_SpdStep, HIGH);
  }
}

void SpinDownPot2() {
  // Fast spin to zero
  digitalWrite(RM_SpdSel, LOW);         // Set outputs to RM digital pot
  digitalWrite(RM_SpdUpDown, LOW);      // Down
  digitalWrite(RM_SpdStep, HIGH);       // Ready to step
  //
  while (analogRead(A1) != 0 {          // Spin pot down to zero
    digitalWrite(LM_SpdStep, HIGH);
    digitalWrite(LM_SpdStep, LOW);
    digitalWrite(LM_SpdStep, HIGH);
  }
}

void RollTheDice() {                    // Randomize the robot's response to sensor inputs
  diceRoll = random(1,7);               // Return a random value from 1-6 (like a die)
  coinToss = random(0,6) % 2;           // Returns 0 (if even: 0,2,4) OR 1 (if odd: 1,3,5)
}

void Fwd_Stop_Rev() {
  // Stop forward motion, pause, enable reverse controls
  digitalWrite(LM_FwdEnable, LOW);      // Disable forward motor controls
  digitalWrite(RM_FwdEnable, LOW);
  delay(500);                           // Give relays time to de-energize
  SpinDownPot1();                       // SPIN speed pots to zero
```

```
  SpinDownPot2();
  digitalWrite(LM_RevEnable, HIGH);     // Enable reverse motor controls
  digitalWrite(RM_RevEnable, HIGH);
}

void Rev_Stop_Fwd() {
  // Stop reverse motion, pause, enable forward controls
  // RAMP pots to zero
  digitalWrite(LM_RevEnable, LOW);      // Disable reverse motor controls
  digitalWrite(RM_RevEnable, LOW);
  delay(500);                           // Give relays time to de-energize
  SpinDownPot1();                       // SPIN speed pots to zero
  SpinDownPot2();
  digitalWrite(LM_FwdEnable, HIGH);     // Enable forward motor controls
  digitalWrite(RM_FwdEnable, HIGH);
}

void RightFrontOnly() {
  // RF
  Fwd_Stop_Rev()
  // VEER reverse to left (LM < RM)
  Rev_Stop_Fwd()
  // PIVOT forward to right (accelerate, LM=on, RM=off)
  // RAMP pots to speed
  // FORWARD straight
}

void RightCenterFront() {
  // RF+CF
  Fwd_Stop_Rev()
  // VEER reverse to left (LM < RM)
  Rev_Stop_Fwd()
  // PIVOT forward to right (accelerate, LM=on, RM=off)
  // RAMP pots to speed
  // FORWARD straight
}

void LeftFrontOnly() {
  // LF
  Fwd_Stop_Rev()
```

```
  // VEER reverse to right (accelerate, LM > RM)
  Rev_Stop_Fwd()
  // PIVOT forward to left (accelerate, LM=off, RM=on)
  // RAMP pots to speed
  // FORWARD straight
}

void LeftCenterFront() {
  // LF+CF
  Fwd_Stop_Rev()
  // REVERSE straight back for a while (accelerate)
  // VEER reverse to left (LM < RM)
  Rev_Stop_Fwd
  // PIVOT forward to right (accelerate, LM=on, RM=off)
  // RAMP pots to speed
  // FORWARD straight
}

void (SideCenterFronts() {
  // CF or LF+RF or all fronts)
  Fwd_Stop_Rev()
  // REVERSE straight back for a while (accelerate)
  Rev_Stop_Fwd()
  /* IF CoinToss == HIGH
        PIVOT forward to left (LM=off, RM=on)
        RAMP pots to speed
     ELSE
        PIVOT forward to right (LM=on, RM=off)
        RAMP pots to speed
     ENDIF
  FORWARD straight
  */
}

void RearOnly() {
  // Rear only
  digitalWrite(LM_RevEnable, LOW);      // Disable reverse motor controls
  digitalWrite(RM_RevEnable, LOW);
  delay(250);                           // Give relays time to de-energize
  SpinDownPot1();                       // SPIN speed pots to zero
```

```
  SpinDownPot2();
  digitalWrite(LM_FwdEnable, HIGH);     // Enable forward motor controls
  digitalWrite(RM_FwdEnable, HIGH);
  /*
  IF CoinToss == HIGH
    VEER forward to left (accelerate, LM < RM)
  ELSE
    VEER forward to right (accelerate LM > RM)
  ENDIF
  FORWARD straight
  RAMP pots to speed
  */
}

void WanderAround() { // GENERAL WANDERING BEHAVIOR

  // PAUSE for a while

  // CHECK and respond to PIR sensors

  // CHECK and respond to PING sensors

  RollTheDice()

  switch (diceRoll) {
    case 1:
      // VEER forward to left (accelerate, LM < RM)
      // RAMP pots to speed
      break;
    case 2:
      // VEER forward to right (accelerate, LM > RM)
      // RAMP pots to speed
      break;
    case 3:
      // SPIN forward to left (accelerate, LM=REV, RM=FWD);
      // RAMP pots to speed
      break;
    case 4:
      // SPIN forward to right (accelerate, LM=FWD, RM=REV);
      // RAMP pots to speed
```

```
      break;
    case 5:
      // PIVOT forward to left (accelerate, LM=off, RM=on)
      // RAMP pots to speed
      break;
    case 6:
      // PIVOT forward to right (accelerate, LM=on, RM=off)
      // RAMP pots to speed
      break;
    default:
  }
}
```