ROCO222: Intro to sensors and actuators

Lecture 4

Arduino interrupts

Procedural programming

- Procedural programs have consisted of a list of statements executed in order
- When that order changed, whether or not to perform certain actions (such as repeat statements in a loop, branch to another statement, or invoke a method) was controlled by the logic of the program

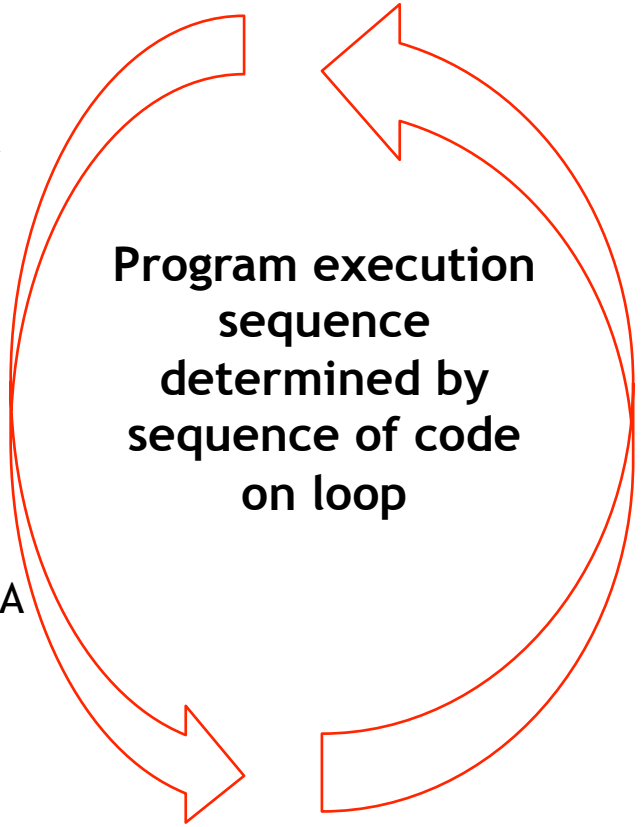
```
void loop(){

  //forward @ full speed
  digitalWrite(12, HIGH); //Establishes forward direction of Channel A
  digitalWrite(9, LOW);   //Disengage the Brake for Channel A
  analogWrite(3, 255);    //Spins the motor on Channel A at full speed
  delay(3000);

  digitalWrite(9, HIGH); //Engage the Brake for Channel A
  delay(1000);

  //backward @ half speed
  digitalWrite(12, LOW); //Establishes backward direction of Channel A
  digitalWrite(9, LOW);   //Disengage the Brake for Channel A
  analogWrite(3, 123);    //Spins the motor on Channel A at half speed
  delay(3000);

  digitalWrite(9, HIGH); //Engage the Brake for Channel A
  delay(1000);
}
```



Event-Driven Programming

In event-driven programming

- objects are created that can fire events
- listener objects are created that can react to the events

- The program itself no longer determines the order in which things can happen

- Instead, the events determine the order

React to button push

```
// specify input contact pin to detect VT closure
// needs to be pin that supports interrupts
// 2,3 for Uno
// 2, 3, 18, 19, 20, 21 for Mega
const int BUTTON_PUSH      = 2;


// callback from button push interrupt
static void buttonPushed()
{
    // do axtion when button pushed here
    this_g->ContactDetected();
}



  // initialize the pushbutton pin as an input:
  pinMode(BUTTON_PUSH, INPUT);

  // trigger when the button pin goes from low to high
  attachInterrupt(digitalPinToInterrupt(BUTTON_PUSH), buttonPushed,
      RISING);
```

Arduino interrupts from a timer

```
#include "TimerOne.h"

// stepper interrupt service routine
void stepperISR()
{
  // call single phase stepper coil update
  stepper.fullStepSinglePhaseCCW();
}

void setup() {

  // setup stepper control
  stepper.Init();

  // setup timer
  Timer1.initialize(3000);
  Timer1.attachInterrupt(stepperISR);
}
```

User defined function
This is the callback function that is called by the timer interrupt

In the Arduino setup function we initialize any parameters
Also we install the timer interrupt service routing

Using an Arduino to read an encoder

```
#include "Encoder.h"

//   Pins connected to your encoder.
//   Best Performance: both pins have interrupt capability
Encoder myEnc(2, 3);
void setup() {
  Serial.begin(9600);
  Serial.println("Basic Encoder Test:");
}

long newPosition = 0;
int ByteReceived;
```

```
void loop()
{
  // reset
  if (Serial.available() > 0)
  {
    // reset
    ByteReceived = Serial.read();
    myEnc.write(0);
  }
  newPosition = myEnc.read();
  Serial.println(newPosition);
}
```

- Arduino example
- Uses Encoder class