Electronics
Real Time Programming

Terry Sturtevant

Wilfrid Laurier University

November 29, 2017

Real Time Programming

Real Time Programming

- Runs continuosly

Real Time Programming

- Runs continuosly
- based on “events”

Real Time Programming

- Runs continuosly
- based on “events”

  e.g. transitions on input pins

Real Time Programming

- Runs continuosly
- based on “events”

  e.g. transitions on input pins

  events can happen at random times and in unpredictable sequences

Real Time Programming

- Runs continuosly
- based on “events”

  e.g. transitions on input pins

  events can happen at random times and in unpredictable sequences
- main program is a software *state machine*.

Sections

Sections

- **definition section**

Sections

- **definition section**

  constants, macros, etc.

Sections

- **definition section**

  constants, macros, etc.
- **setup section**

Sections

- **definition section**

  constants, macros, etc.
- **setup section**

  code to be run once

Sections

- **definition section**

  constants, macros, etc.
- **setup section**

  code to be run once
- **infinite loop section**

Sections

- **definition section**

  constants, macros, etc.
- **setup section**

  code to be run once
- **infinite loop section**

  repeated until shut down

Arduino Serial sample code

Arduino Serial sample code

```
void setup() {
  Serial.begin(9600);
  while (!Serial) {
    ;
  }
}

void loop() {
  if (Serial.available() > 0) {
    inByte = Serial.read();
    Serial.write(inByte);
    }
}
```

PySpidev sample code

PySpidev sample code

```
import spidev
spi = spidev.SpiDev()
spi.open(0,0) #port,device
# use port 0, chip select CE0
while True:
   strval = raw_input("val (0...255, q=quit):")
   if strval == 'q':
      break
   else:
      value = int(strval)
      dummy = spi.xfer2([49,value])
spi.close()
```

Real-time sequencing options

Real-time sequencing options

1. **polling**

Real-time sequencing options

1. **polling**

   check to see if task done

Real-time sequencing options

1. **polling**

   check to see if task done

2. **timed**

Real-time sequencing options

1. **polling**

   check to see if task done

2. **timed**

   allow fixed time for each task

Real-time sequencing options

1. **polling**

   check to see if task done
2. **timed**

   allow fixed time for each task
3. **interrupts**

Real-time sequencing options

1. **polling**

   check to see if task done

2. **timed**

   allow fixed time for each task

3. **interrupts**

   use events to indicate task completion

Real-time sequencing options

1. **polling**

   check to see if task done
2. **timed**

   allow fixed time for each task
3. **interrupts**

   use events to indicate task completion
4. **capture/compare**

Real-time sequencing options

1. **polling**

   check to see if task done

2. **timed**

   allow fixed time for each task

3. **interrupts**

   use events to indicate task completion

4. **capture/compare**

   tasks run at intervals, completion indicated by event

Real-time sequencing options

1. **polling**

   check to see if task done
2. **timed**

   allow fixed time for each task
3. **interrupts**

   use events to indicate task completion
4. **capture/compare**

   tasks run at intervals, completion indicated by event

   probably includes interrupts

- These are not mutually exclusive.

- These are not mutually exclusive.
- The first two make tasks sequential, the last two create a software state machine.

- These are not mutually exclusive.
- The first two make tasks sequential, the last two create a software state machine.
- A software state machine can be prioritized or round-robin.

- These are not mutually exclusive.
- The first two make tasks sequential, the last two create a software state machine.
- A software state machine can be prioritized or round-robin.
- The first two make the *response to system events* depend on *program structure*.

- These are not mutually exclusive.
- The first two make tasks sequential, the last two create a software state machine.
- A software state machine can be prioritized or round-robin.
- The first two make the *response to system events* depend on *program structure*.
- The last two make the *program structure* depend on *response to system events*.