Lab Session and Coursework

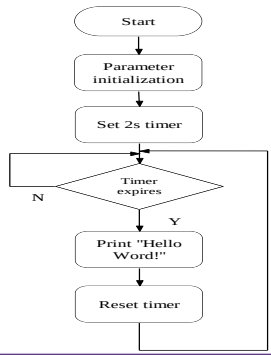
Computer Science Department
Loughborough University (UK)

Outline

- Six Sample Applications
- Coursework Design

Sample 1: Timer Setting

- Understand the operations of the SensiNode
- Learn the usage of "Timer" and "Print" function



Sample 1: Timer Setting

- Parameter initializations

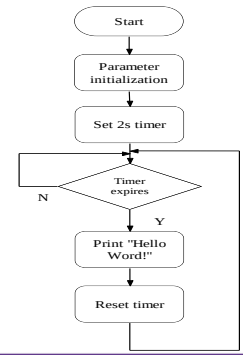
```
static struct etimer timer;
```

- Set 2 s timer

```
etimer_set(&timer, CLOCK_CONF_SECOND * 2);
```

- Timer expires, print, reset timer

```
while (1)
{
PROCESS_WAIT_EVENT_UNTIL
 (ev == PROCESS_EVENT_TIMER);
    printf("Hello World!\r\n");
    etimer_reset(&timer);
}
```

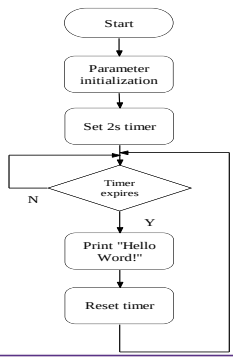


Sample 1: Timer Setting

- Summary for sample 1:

1. An application is an endless loop

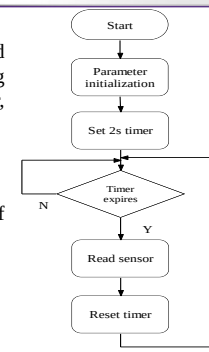
2. Basic procedures are: initialize parameters, execute regular operations if certain criteria is satisfied



Sample 2: Reading Sensor

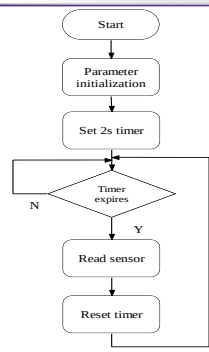
- A number of sensors are equipped with each sensinode, including temperature sensor, light sensor, accelerometer, voltage sensor.
- Difference:

Execute "Read sensor" instead of "Print Hello Word!"



Sample 2: Reading Sensor

- Read temperature sensor
  sensor->value(ADC_SENSOR_TYPE_TEMP);
- Read accelerometer
  sensor->value(ADC_SENSOR_TYPE_ACC_X);
  sensor->value(ADC_SENSOR_TYPE_ACC_Y);
  sensor->value(ADC_SENSOR_TYPE_ACC_Z);
- Read light sensor
  sensor->value(ADC_SENSOR_TYPE_LIGHT);
- Read voltage
  sensor->value(ADC_SENSOR_TYPE_VDD);



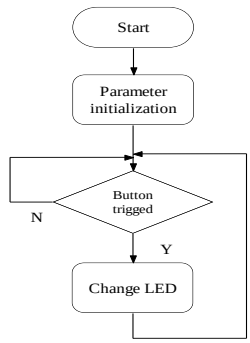
Sample 3: Interruption Handling

- Interruption is usually caused by the low level hardware which requires the processor to process
- Button interruption: electrical level

Sample 3: Interruption Handling

- Parameter initialization
  struct sensors_sensor *sensor;
- Interruption judgment

```
while (1)
{
    PROCESS_WAIT_EVENT_UNTIL
    (ev== sensors_event);

    if(sensor == &button_1_sensor) {
    leds_toggle(LEDS_GREEN);
    }
}
```

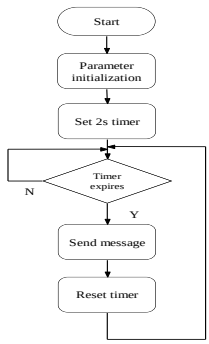


Sample 4: Radio Communication

- Basic communication infrastructure of WSN: Point to point communication
- A node sends out a message
- One node receives the message (unicast)
- All nodes nearby receive the message (broadcast)
- Sender, and receiver

Sample 4: Radio Communication

- Sender



Sample 4: Radio Communication

- Sender: send message

1. **Initialize parameter**

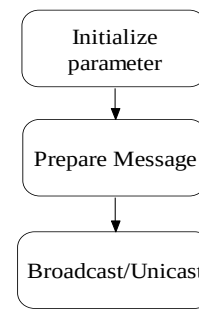
```
rimeaddr_t    addr;
broadcast_open(&bc, 128, &bc_cb);
unicast_open(&uc, 129, &unicast_callbacks);
```

2. **Prepare message body**

```
packetbuf_copyfrom("Hello        everyone", 14);
```

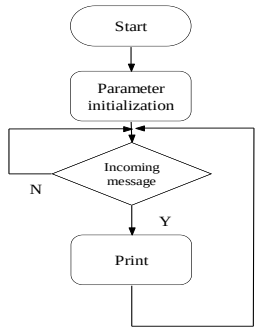
3. **Call sending function**

```
Broadcast: broadcast_send(&bc)
Unicast:   addr.u8[0] = 0xff;
           addr.u8[1] = 0xff;
           unicast_send(&uc, &addr);
```



Sample 4: Radio Communication

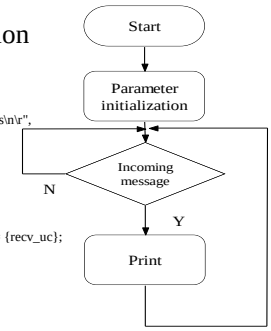
- Receiver
- Parameter initialization
  Callback registration



Sample 4: Radio Communication

- Unicast callback registration

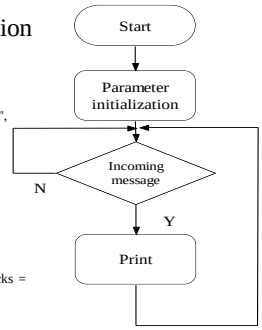
```
static void
recv_uc(struct unicast_conn *c, rimeaddr_t *from)
{
   printf("unicast   from %02x.%02x len = %d buf = %s\n\r",
                        from->u8[0],
                        from->u8[1],
                        packetbuf_datalen(),
                        (char *)packetbuf_dataptr());
}
static const struct unicast_callbacks unicast_callbacks = {recv_uc};
```



Sample 4: Radio Communication

- Broadcast callback registration

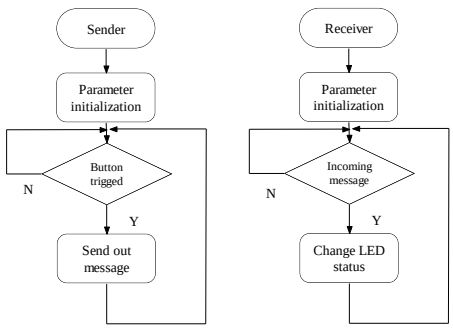
```
static void
recv_bc(struct broadcast_conn *c, rimeaddr_t *from)
{
  uint16_t rssi;
  printf("broadcast from %02x.%02x len = %d buf = %s",
                        from->u8[0],
                        from->u8[1],
                        packetbuf_datalen(),
                        (char *)packetbuf_dataptr());

rssi = packetbuf_attr(PACKETBUF_ATTR_RSSI);
  printf("  RSSI=%d\n\r",rssi);
  packetbuf_clear();
}
static const struct broadcast_callbacks broadcast_callbacks =
{recv_bc};
```



Sample 5: Remote Control

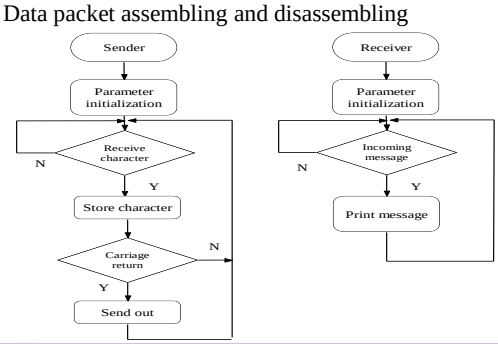
- A sensinode sends out message when a button is trigged
- Another sensinode changes the statues of the LED if a message is received
- Sample 3 (Interruption handlling) + Sample 4 (Radio communication)

Sample 5: Remote Control



Sample 6: Data Processing

- Data packet assembling and disassembling



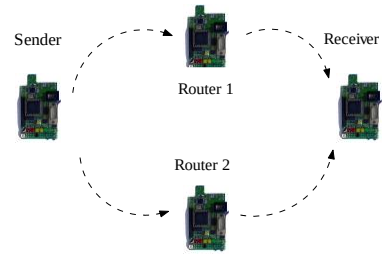
Sample 6: Data Processing

- Store character

```
PROCESS_WAIT_EVENT_UNTIL(ev == serial_line_event_message);
databuffer[counter] = (char)(*(int *)data);

 if(databuffer[counter]==13)
    {
         packetbuf_copyfrom(databuffer, counter);
          addr.u8[0] = 0xff;
        addr.u8[1] = 0xff;
        unicast_send(&uc, &addr);
        counter=0;
    }else{
        counter++;
}
```

Coursework

- Previous sample: point to point communication
- Coursework: multi-hop communication

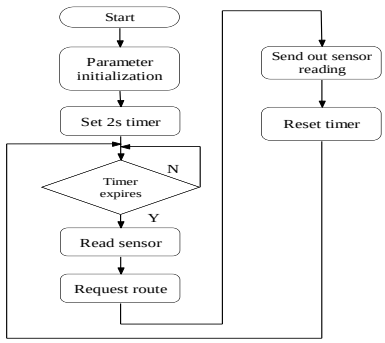


Coursework

- Sender

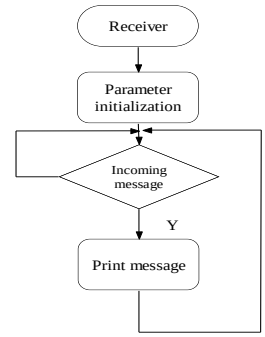
Read sensor: sample 2

Send out message:
sample 4 + sample 6



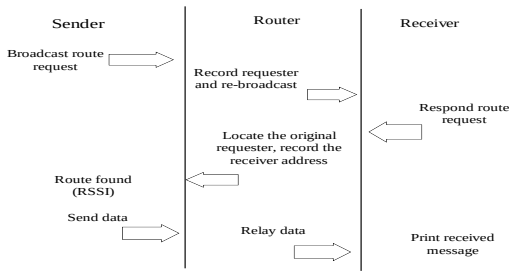
Coursework

- Receiver



Coursework

- Routing protocol: route request



Thanks