Building Real-Time Embedded Applications on QduinoMC

A Web-connected 3D Printer Case Study

Zhuoqun (Tom) Cheng, Richard West and Ying Ye
3D Printing HOW-TO

- **CAD Model**
- **Slicing Program**
- **G-code**
- **Firmware**
- **Printed Object**
- **Extruder**
- **Table**

**Steps**:
1. **Melt the filament**
2. **Move in 3D space**
3. **Deposit onto the table**

**Serial**
Web-connected 3D Printer
Web-connected 3D Printer

Remote Job Submission
Local Slicing
Correctness Verification
### Microprocessor

- **Atmel AVR, 8 bit, 20 MHz**

### SRAM
- **8 KB**

### I/O
- **UART, SPI, I2C, PWM, GPIO**
Custom Controller

- RAMPS shield: I/O extension board
- Intel MinnowBoard MAX
- 64 bit dual-core Atom E3825 1.33 GHz
- 2 GB memory
- 86 GPIOs, I2C, SPI, UART, PWM

RAMPS shield: I/O extension board

Intel MinnowBoard MAX
Custom Controller

RAMPS shield

Companion Analog Circuits

MinnowMAX
Marlin Firmware

Main loop

Read G-code

Translate to motor rotation

Temperature Control PID

G-code

Timer1 Interrupt

Interpret Motor Data

Timer2 Interrupt

Adjust fan & heater

Temperature

PID output

Data

Variable Period

8ms Period

Extruder

Motor
### Original Marlin

- Main loop + interrupts handlers
- Timer interrupts
- AVR I/O instructions
- All computations in the main loop

### Jitter of the extruder, when submitting relatively large files

### Is this bad? Why?

![Diagram showing Marlin, lighttpd, Spooler, Linux, MinnowMAX](image-url)
The Timing Problem

$$Volume = \gamma \cdot T = L \cdot H \cdot d$$

$$= f \cdot T \cdot S \cdot H \cdot d$$

$$H = \frac{\gamma}{S \cdot d} \cdot \frac{1}{f}$$

f -- pulse frequency
S -- linear displacement per pulse
struct timespec period = {.tv_sec = 0, .tv_nsec = 100000};
while (1) {
    nanosleep(&period, NULL);  /* sleep for 100 us */
    mraa_gpio_write(GPIO6, HIGH); /* write 1 to gpio6 */
    mraa_gpio_write(GPIO6, LOW);  /* write 0 to gpio6 */
}

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theoretical</td>
<td>10 kHz</td>
<td>100000 ns</td>
</tr>
<tr>
<td>Measured</td>
<td>7.91 kHz</td>
<td>100000 ns + 26422 ns</td>
</tr>
<tr>
<td>Original PrintrBoard</td>
<td>9.96 kHz</td>
<td>100000 ns + 401 ns</td>
</tr>
</tbody>
</table>
struct timespec period = {.tv_sec = 0, .tv_nsec = 100000};

while (1) {
    nanosleep(&period, NULL); /* sleep for 100 us */
    mraa_gpio_write(GPIO6, HIGH); /* write 1 to gpio6 */
    mraa_gpio_write(GPIO6, LOW); /* write 0 to gpio6 */
}
struct timespec period = {.tv_sec = 0, .tv_nsec = 100000};

while (1) {
    nanosleep(&period, NULL);  /* sleep for 100 us */
    mraa_gpio_write(GPIO6, HIGH); /* write 1 to gpio6 */
    mraa_gpio_write(GPIO6, LOW); /* write 0 to gpio6 */
}

### Real-time environment setup

- PREEMPT_RT patch
- Setting scheduling priority
- Locking pages into memory

### Further optimization

- Shield a core from the scheduler: isolcpu, cset...
- Disable timer interrupt on a core: CONFIG_NO_HZ_FULL

### Lack of a simple and uniform programming interface
QduinoMC

Goals

<table>
<thead>
<tr>
<th>Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to use</td>
</tr>
<tr>
<td>Easy to port existing Arduino program</td>
</tr>
<tr>
<td>Take advantage of the multi-core</td>
</tr>
<tr>
<td>Allow QoS specification</td>
</tr>
<tr>
<td>Low I/O access overhead</td>
</tr>
</tbody>
</table>

loop (loopID, budget, period, [coreID])

noInterrupts (device, coreID)

noTimer (coreID)

interruptsVcpu (device, budget, period, [coreID])
digitalWrite () / digitalRead ()
Marlin on QduinoMC

loop (1, 10, 100, 1), loop (2, 30, 100, 0), loop (3, 1, 80, 0)

interruptsVCPU (I2C, 10ms, 100ms), interruptsVCPU (NIC, 10ms, 100ms)

noTimer (1), noInterrupts (ALL, 1)

Web server / Spooler -- default
void setup () {
    pinMode(GPIO6, OUTPUT);
    noInterrupts(ALL, 1); noTimer(1);
}

void loop (1, 100, 100, 1) {
    delayBusyNanoseconds(100000);
    digitalWrite(GPIO6, 1); digitalWrite(GPIO6, 0);
}

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theoretical</td>
<td>10 kHz</td>
<td>100000 ns</td>
</tr>
<tr>
<td>Measured QduinoMC</td>
<td>9.569 kHz</td>
<td>100000 ns + 4504 ns</td>
</tr>
<tr>
<td>Measured Linux</td>
<td>7.91 kHz</td>
<td>100000 ns + 26422 ns</td>
</tr>
<tr>
<td>Original PrintrBoard</td>
<td>9.96 kHz</td>
<td>100000 ns + 401 ns</td>
</tr>
</tbody>
</table>
Test Object

- Higher quality
- Faster printing
- 10% code size reduction
- Intuitive and clear code structure
Conclusion & Future Work

- Designed and built a platform to ease the development of IoT applications with critical timing requirements.

- Built a web-connected 3D printer as a case study and analyzed 3D printers’ real-time properties.

- Future work will extend web connectivity to support local slicing and print verification.
Thank you!

Questions?