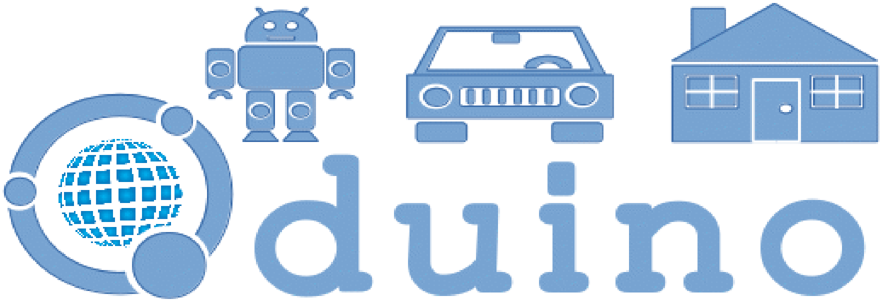
Qduino: A Multithreaded Arduino System for Embedded Computing

Zhuoqun Cheng, Ye Li, Richard West



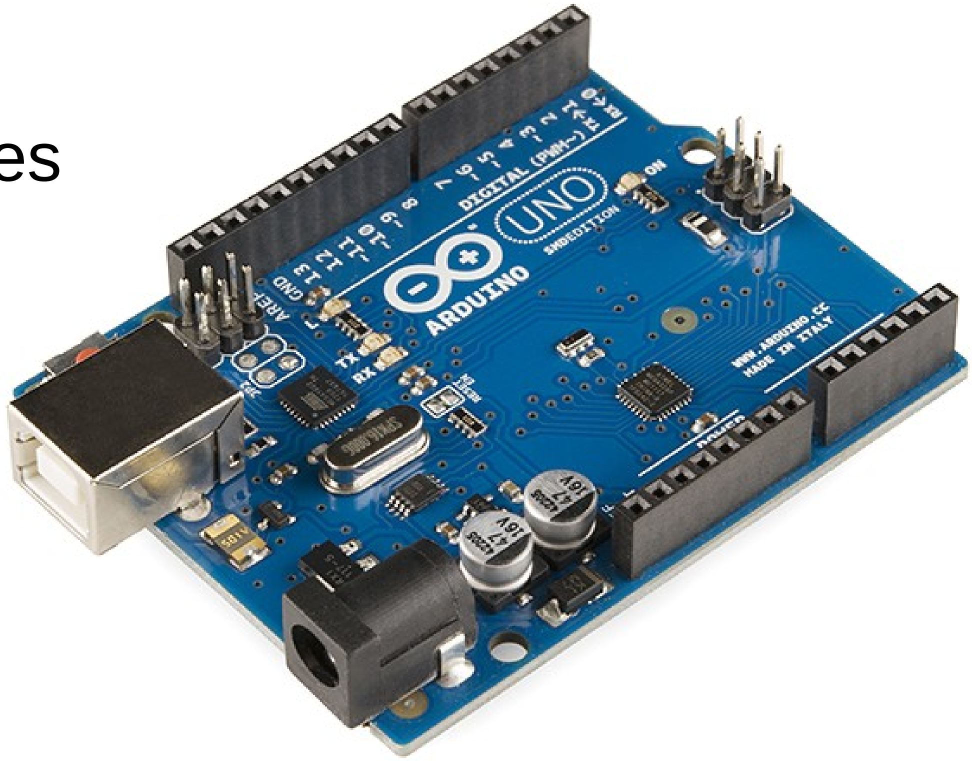
Computer Science

Background

- Many Robotics, Internet of Things, Home Automation applications have been developed recently
  - Perform complicated computing tasks
  - Interact with the physical world
- Need an easy-to-use platform to develop applications
  - High processing capabilities
  - Straightforward hardware and software interface

Background

- Arduino
  - Digital and analog GPIOs
  - Simple API
  - Low processing capabilities
    - Arduino Uno: 16MHz 8-bit ATmega328P



Background

- More powerful Arduino-compatible boards emerge to meet the demands
  - Intel Galileo: 400MHz Intel Quark X1000
  - Intel Edison: 500MHz dual-core Atom
  - Arduino-compatible: the same GPIO layout with the standard Arduino boards

Background

- The standard Arduino runs sketches (Arduino program) on the bare metal
- New boards are shipped with Linux
  - Able to afford the overhead of operating systems
  - To cope with the complexity of the hardware
  - Run sketches as Linux processes

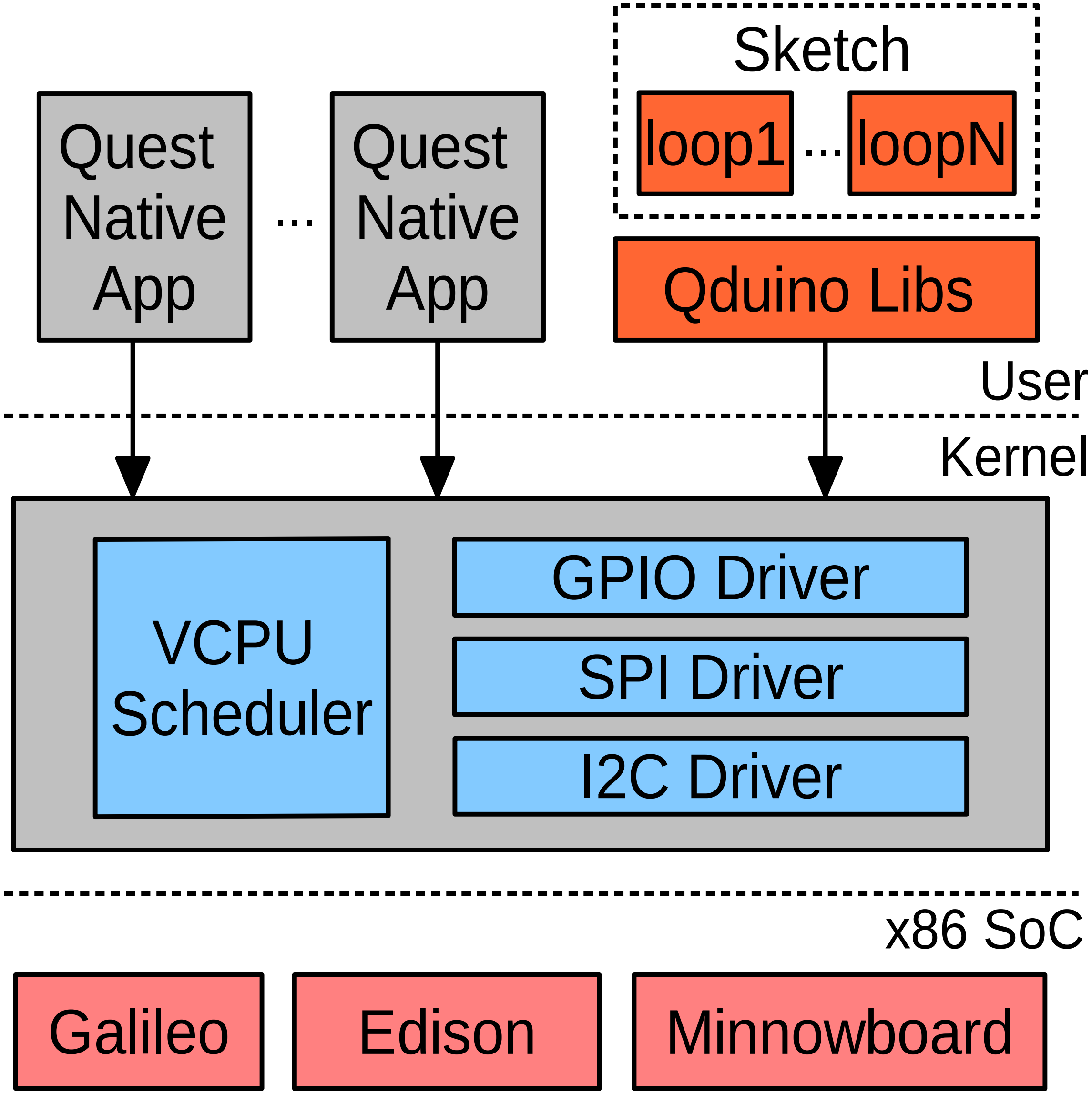
Motivation

- Linux lacks predictability
  - Many embedded applications have real-time requirements
  - RTOS is needed
- The standard Arduino API designed for a single thread of execution
  - No multithreading or concurrency
  - Fails to utilize computing resources and hardware parallelism

Contributions

- Qduino: a programming environment that provides support for preemptive multithreading Arduino API that guarantees timing predictability of different control flows in a sketch
  - Multithreaded sketches, and synchronization and communication between control flows
  - Temporal isolation between different control flows and asynchronous system events, e.g., interrupts
  - Predictable event delivery for I/O handling in sketches

Qduino Architecture



Arduino vs Qduino APIs

Category	Standard APIs	New APIs (backward compatible)
Structure	setup(), loop()	loop(id, C, T)
Digital and Analog I/Os	pinMode(), digitalWrite(),digitalRead(), anlogWrite(), anlogRead()	
Interrupts	Interrupts(), noInterrupts(), attachInterrupt(pin, ISR, mode), detachInterrupt(pin)	interruptsVcpu(C, T), attachInterruptVcpu(pin, ISR, mode, C, T)
Synchronization & Communication		spinlock, four-slot channel, ringbuffer
Other Utility Functions	micros(), delay(), min(), sqrt(), sin(), isLowerCase(), random(), bitset(), ...	

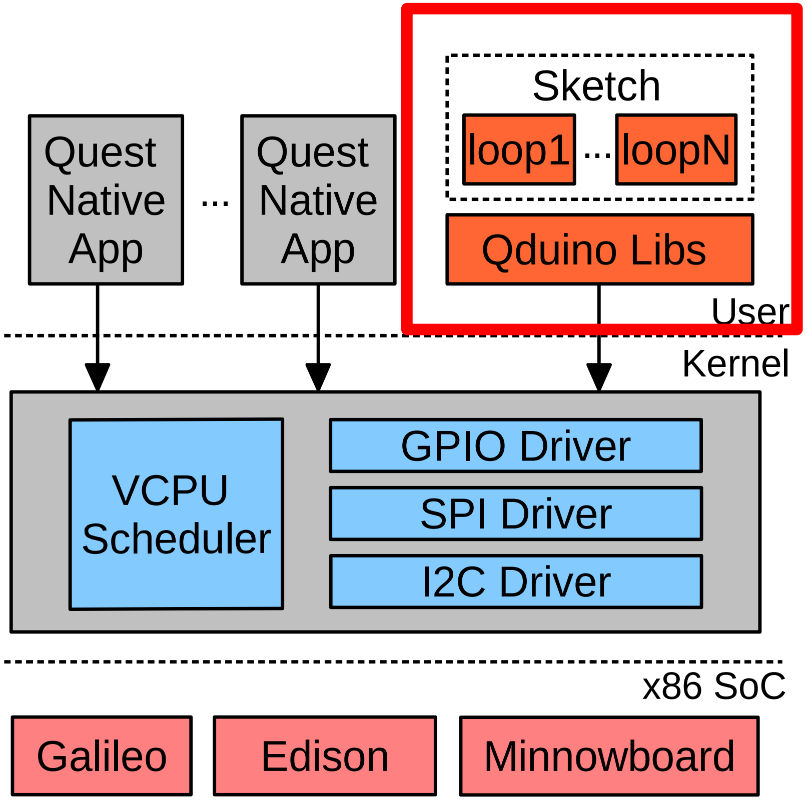
Contributions

- Qduino:
  - Multithreaded sketches, and synchronization and communication between control flows
  - Temporal isolation between different control flows and asynchronous system events, e.g., interrupts
  - Predictable event delivery for I/O handling in sketch

Multithreaded Sketch

Structure	loop(), setup()	loop(id, C, T)

- Standard API
  - Only one loop() is allowed
  - Blocking I/Os block the sketch
- Qduino:
  - Up to 32 loop() in one sketch
  - Each loop() function is assigned to a Quest thread



Multithreaded Sketch

- Benefits
  - Loop interleaving
    - Blocking I/Os won't block the entire sketch
    - increase CPU utilization
  - Easy to write sketches with parallel tasks
    - Example: toggle pin 9 every 2s, pin 10 every 3s

Multithreaded Sketch

```
//Sketch 1: toggle pin 9 every 2s
int val9 = 0;

void setup() {
        pinMode(9, OUTPUT);
}

void loop() {
        val9 = !val9; //flip the output value
        digitalWrite(9, val9);
        delay(2000); //delay 2s
}
```

```
//Sketch 2: toggle pin 10 every 3s
int val10 = 0;

void setup() {
        pinMode(10, OUTPUT);
}

void loop() {
        val10 = !val10; //flip the output value
        digitalWrite(10, val10);
        delay(3000); //delay 3s
}
```

Delay(?)

No way to merge them!

Multithreaded Sketch

- Inefficient
- Do scheduling by hand
  - Hard to scale

```
int val9, val10 = 0;

int next_flip9, next_flip10 = 0;

void setup() {
        pinMode(9, OUTPUT);
        pinMode(10, OUTPUT);
}

void loop() {
        if (millis() >= next_flip9) {
                val9 = !val9; //flip the output value
                digitalWrite(9, val9);
                next_flip9 += 2000;
        }
        if (millis() >= next_flip10) {
                val10 = !val10; //flip the output value
                digitalWrite(10, val10);
                next_flip10 += 3000;
        }
}
```

Multithreaded Sketch

- Multithreaded Sketch in Qduino

```
int val9, val10 = 0;
int C = 500, T = 1000;

void setup() {
        pinMode(9, OUTPUT);
        pinMode(10, OUTPUT);
}

void loop(1, C, T) {
        val9 = !val9; //flip the output value
        digitalWrite(9, val9);
        delay(2000);
}

void loop(2, C, T) {
        val10 = !val10; //flip the output value
        digitalWrite(10, val10);
        delay(3000);
}
```

Communication & Synchronization

- Loops – threads
  - Communication via global variables
- Serialized global variable access
  - Explicit: spinlock
  - Implicit: channel, ring buffer

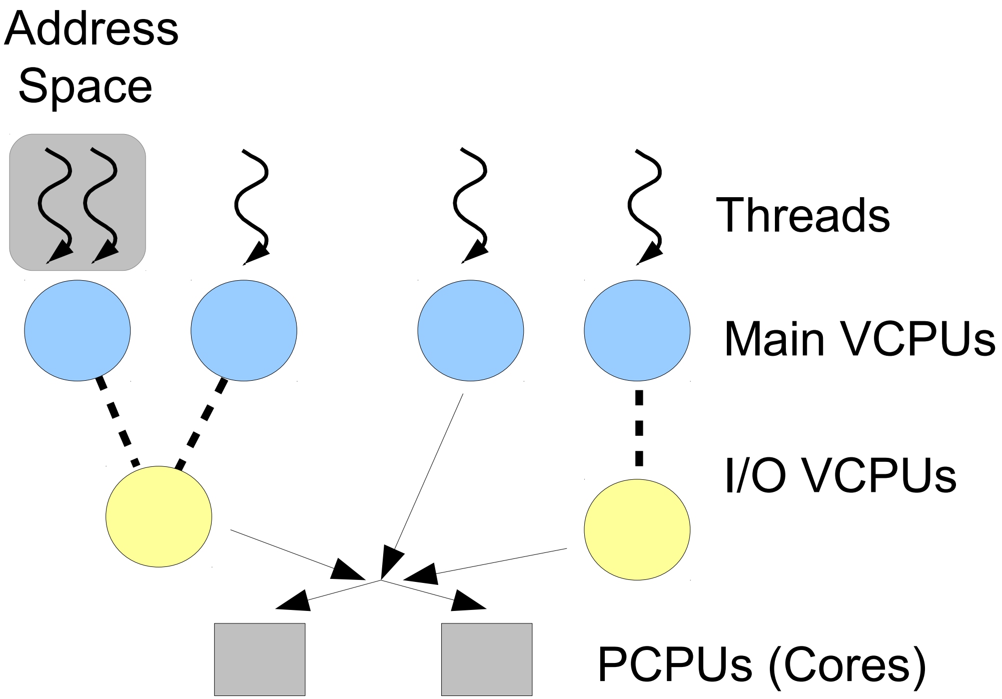
Function Signatures	Category
•spinlockInit(lock) •spinlockLock(lock) •spinlockUnlock(lock)	Spinlock
•channelWrite(channel,item) •item channelRead(channel)	Four-slot
•ringbufInit(buffer,size) •ringbufWrite(buffer,item) •ringbufRead(buffer,item)	Ring buffer

Contributions

- Qduino:
  - Multithreaded sketches, and synchronization and communication between control flows
  - Temporal isolation between different control flows and asynchronous system events, e.g., interrupts
  - Predictable event delivery for I/O handling in sketch

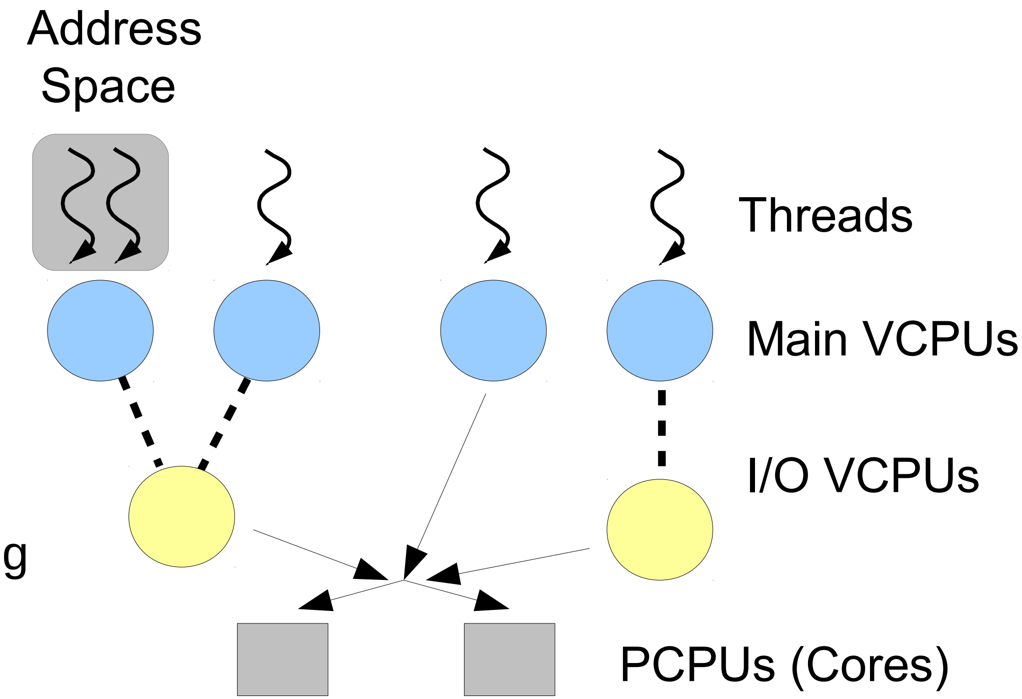
Temporal Isolation

- Real-time Virtual CPU (VCPU) Scheduling
  - VCPU: kernel objects for time accounting and scheduling
  - Two classes:
    - Main VCPU – conventional thread
    - I/O VCPU – threaded interrupt handler



Temporal Isolation

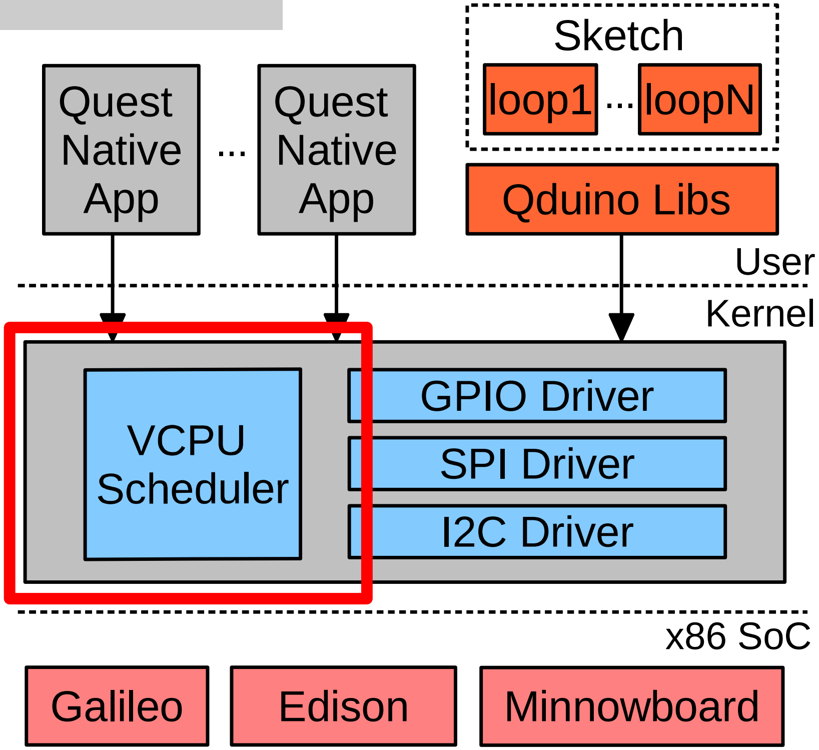
- Real-time Virtual CPU (VCPU) Scheduling
  - Each VCPU has a max budget *C*, a period *T* and a utilization *U* = *C* / *T*
  - Integrate the scheduling of tasks & I/O interrupts
    - Extension to rate-monotonic scheduling
    - Ensure temporal isolation if the Liu-Layland utilization bound is satisfied



Temporal Isolation

Structure	loop(), setup()	loop(id, C, T)
Interrupts	interrupts()	interruptsVcpu(C, T)

- Loop – thread – Main VCPU
  - Specify loop timing requirements
- GPIO interrupt handler – I/O VCPU
  - Control # of interrupts to handle
- Balance CPU time between tasks, as well as tasks and interrupts



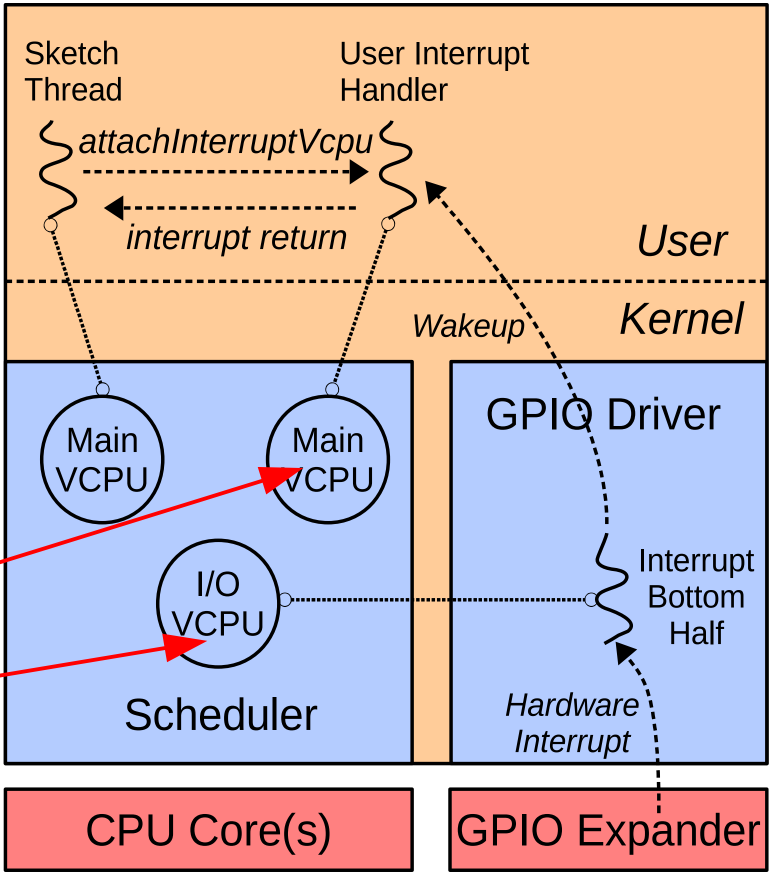
Contributions

- Qduino:
  - Multithreaded sketches, and synchronization and communication between control flows
  - Temporal isolation between different control flows and asynchronous system events, e.g., interrupts
  - Predictable event delivery for I/O handling in sketch

Predictable Events

Category	Standard APIs	Newly added APIs
Interrupts	Interrupts(), noInterrupts(), attachInterrupt(pin, ISR, mode), detachInterrupt(pin)	interruptsVcpu(C, T), attachInterruptVcpu(pin, ISR, mode, C, T)

- Event delivery time: the time interval between the invocation of the ISR and the invocation of the user-level interrupt handler
- Predictable end-to-end event delivery
- *attachInterruptVcpu(..., C, T), interruptsVcpu(C, T)*



Predictable Events

- I/O VCPU (*C*io, *T*io) – threaded interrupt bottom half
- Main VCPU (Ch, Th) – threaded user interrupt handler
- Worst Case Event Delivery Time:

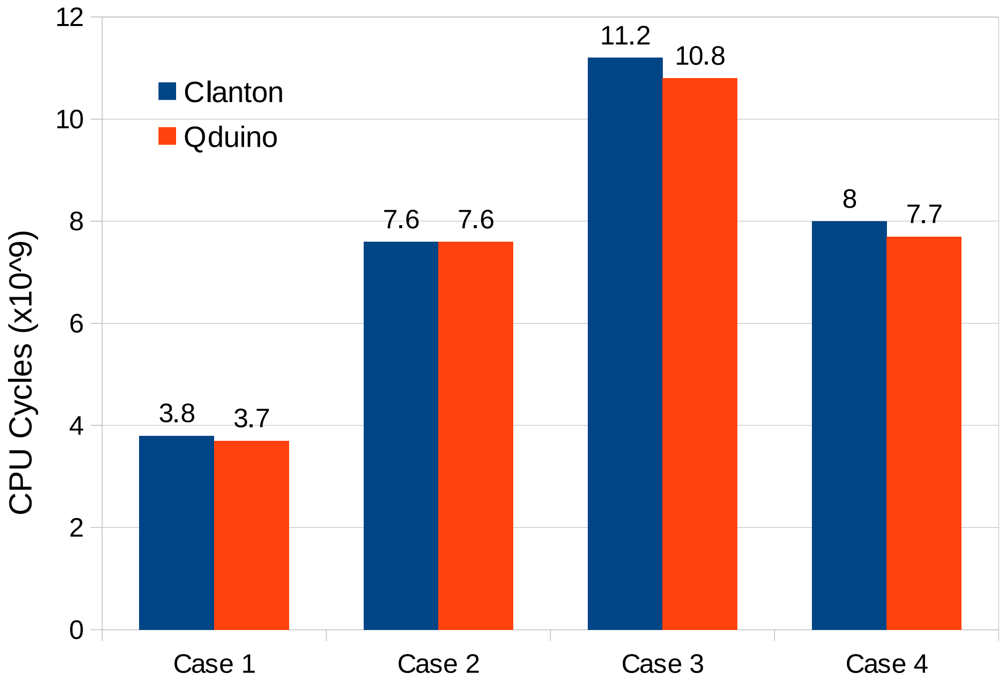
$$\Delta_{WCD}=\Delta_{bh}+(T_h-C_h)=\overset{\text{I/O VCPU used up budget}}{\boxed{(T_{io}-C_{io})}}+\overset{\text{Interrupt bottom half execution time}}{\boxed{\left\lceil\frac{\delta_{bh}}{C_{io}}-1\right\rceil\cdot T_{io}+\delta_{bh}\,mod\,C_{io}}}+\overset{\text{Main VCPU used up budget}}{\boxed{(T_h-C_h)}}$$

Evaluation

- Experiment Setup
  - Intel Galileo board Gen 1
  - Qduino vs. Clanton
    - Clanton Linux 3.8.7 is shipped with the Galileo board

Evaluation

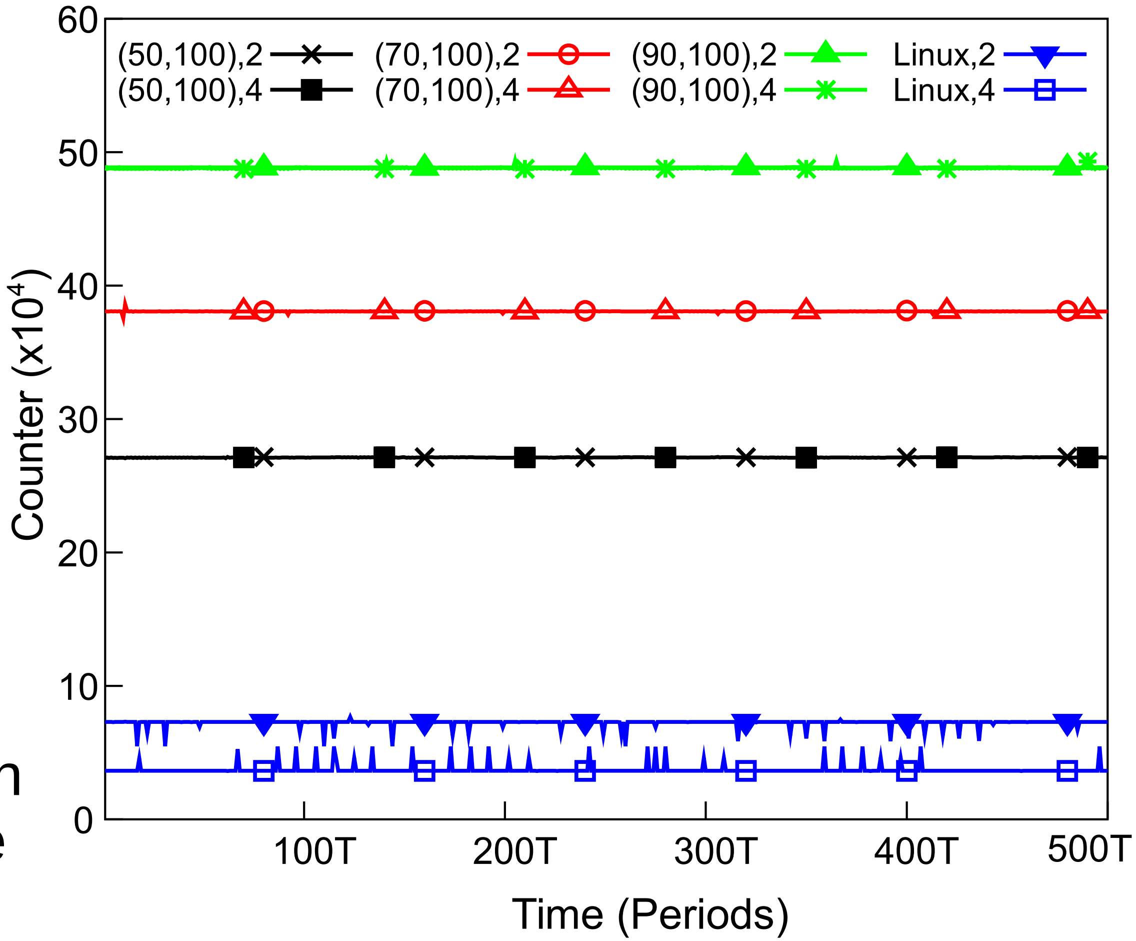
- Multithreaded Sketch
  - Computation-intensive: find all prime numbers smaller than 80000
  - I/O-intensive: 2000 digital write
  - Reduce 30% CPU Cycles



Case #	Description
Case 1	Single-loop digitalWrite()
Case 2	Single-loop findPrime
Case 3	Single-loop digitalWrite() + findPrime
Case 4	Multi-loop digitalWrite() + findPrime

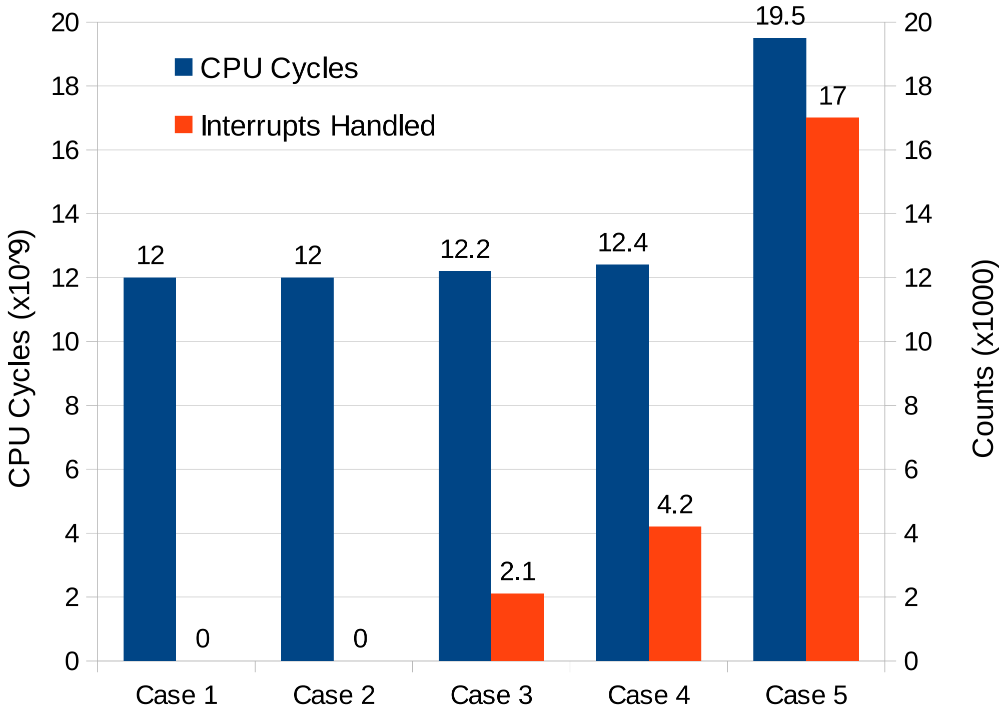
Evaluation

- Predictable loop execution
  - 1 Foreground loop increments a counter during its loop period
  - 2/4 background loops act as potential interference
  - Result interpretation
    - Overlapped – temporal isolation
    - Straight line – timing guarantee



Evaluation

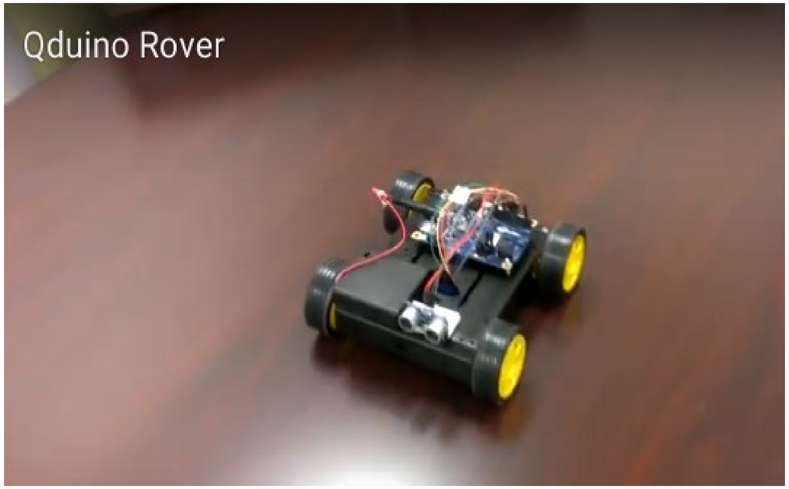
- Temporal Isolation between loops and interrupts
  - Use an external device to toggle pin 2 of Galileo
  - Run findPrime at the same time
  - Execution time of findPrime and # of interrupts handled



Case #	I/O VCPU	External Interrupts
Case 1	10/100	OFF
Case 2	0/100	ON
Case 3	5/100	ON
Case 4	10/100	ON
Case 5	Disabled	ON

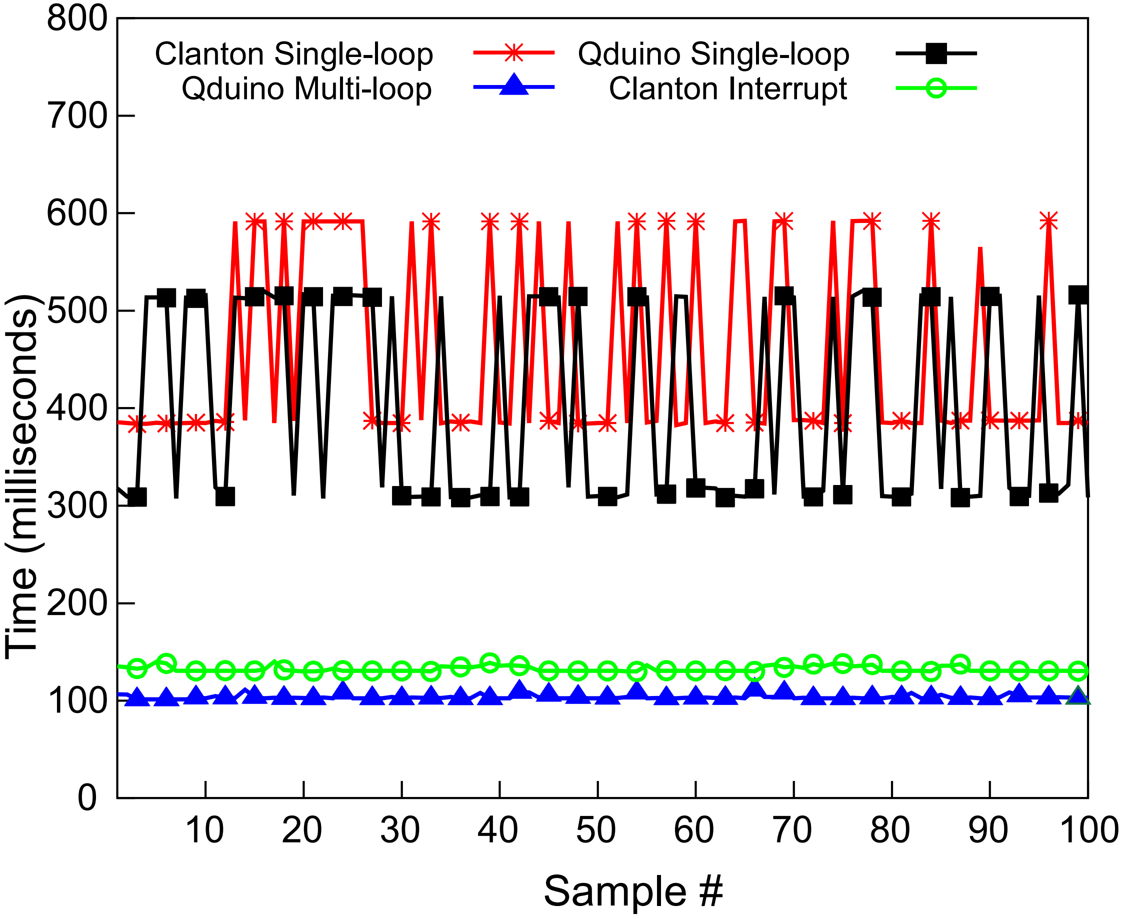
Evaluation

- Autonomous Vehicle
  - Collision avoidance using ultrasonic sensor
  - Two tasks:
    - A sensing task detects distance to an obstacle - delay(200)
    - An actuation task controls the motors - delay(100)



Evaluation

- Autonomous Vehicle
  - Measure the time interval between two consecutive calls to the motor actuation code
  - Clanton single loop
    - delay from both sensing and actuation task
  - Qduino multi-loop
    - No delay from the sensing loop
    - No delay from sensor timeout
  - The shorter the worst case time interval, the faster the vehicle can drive



Conclusions

- Supported Quest RTOS on Intel Arduino-compatible boards
- Designed and implemented an extension to the Arduino API for Quest on new powerful Arduino-compatible boards
  - Multi-loop sketches
  - Real-time guarantee

Thank you!

- Questions?
- More information can be found at:
    - https://www.cs.bu.edu/~richwest/Qduino.php

Future Work

- Conditional loops
- Communication between loops with loop IDs
- Multi-sketches

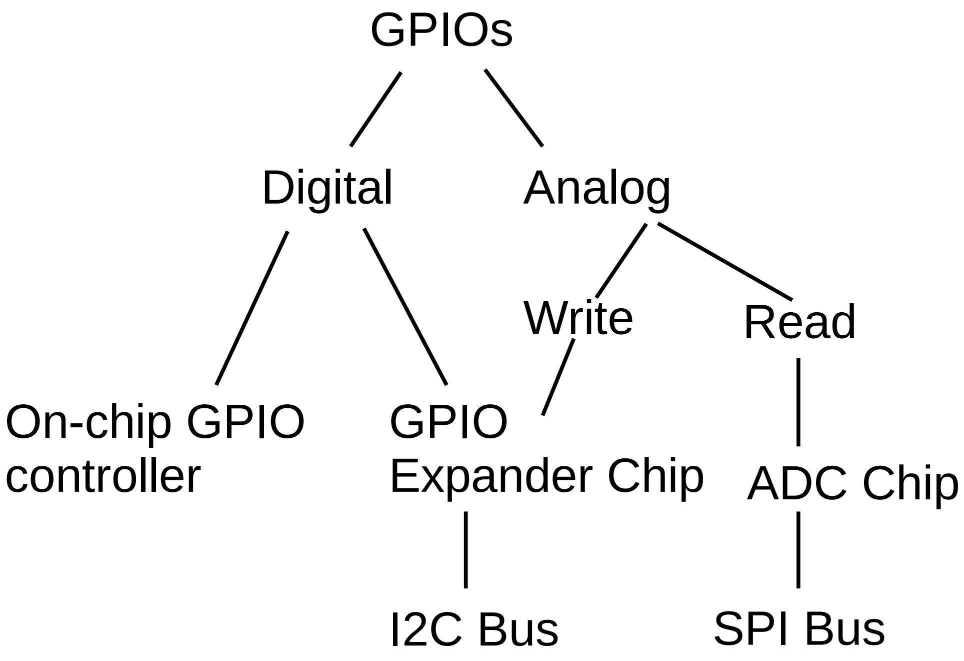
Memory Footprint

	Text (Bytes)	Data (Bytes)
Qduino kernel	953358	321516
Clanton kernel	4390436	336104
Qduino autonomous vehicle sketch	4832	2360
Clanton autonomous vehicle sketch	26249	27652

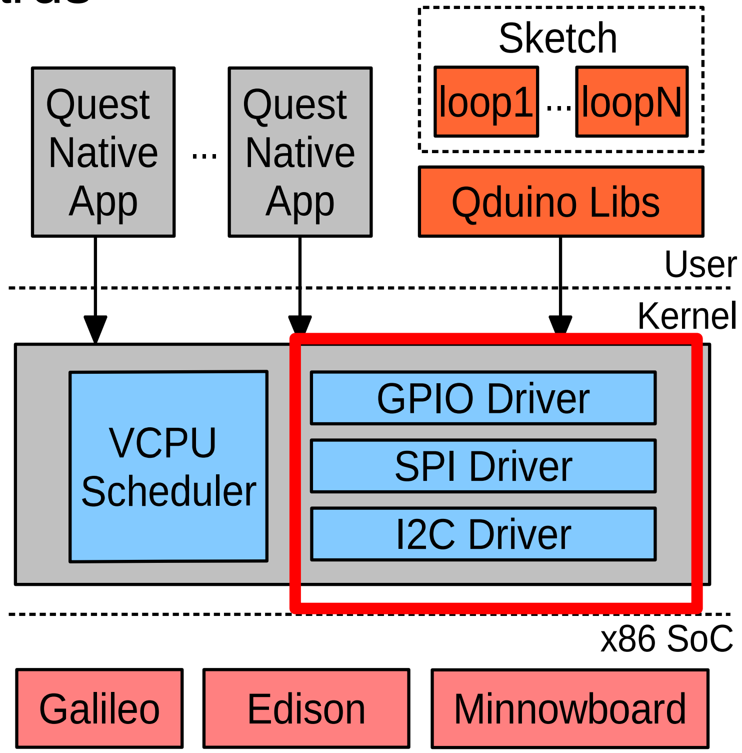
GPIOs

Category	Standard APIs	Newly added APIs
Digital and Analog I/Os	PinMode(), digitalWrite(), digitalRead(), anlogWrite(), anlogRead()	

- Complicated I/O Architecture on new boards



GPIOs
Digital
Analog
Write
Read
On-chip GPIO controller
GPIO Expander Chip
ADC Chip
I2C Bus
SPI Bus



Sketch
loop1 ... loopN
Qduino Libs
Quest Native App ... Quest Native App
User
Kernel
VCPU Scheduler
GPIO Driver
SPI Driver
I2C Driver
x86 SoC
Galileo
Edison
Minnowboard