HATBED: A Distributed Hardware Assisted Testbed for Non-invasive Profiling of IoT Devices

Li Yi, ***Junyan Ma***, *Te Zhang*

School of Information Engineering, Chang'an University, P.R.China

Outline

1 Motivation

2 Non-invasive profiling

3 HATBED Design

4 Preliminary Evaluation

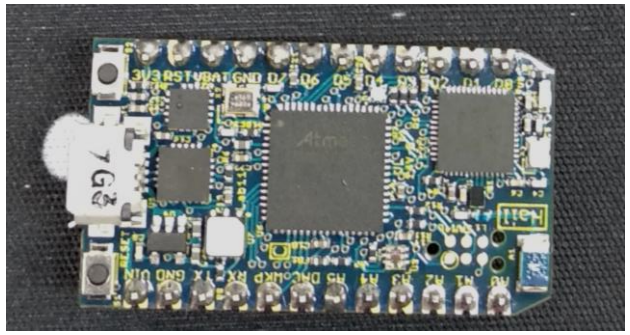
5 Conclusion

Motivation

Motivation

Deeply coupled with the physical world

Difficult to maintain and diagnosis after deployment



Node

Embedded into

Physical World

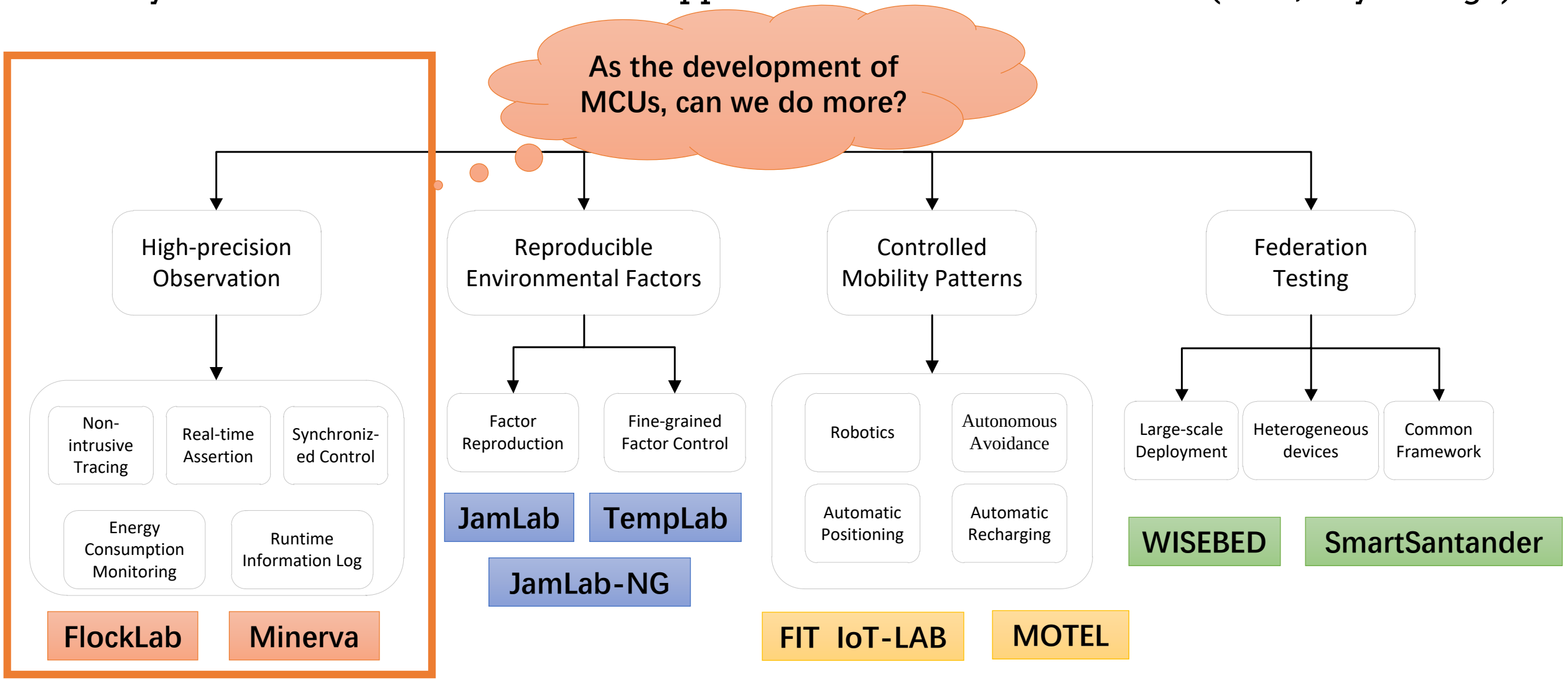
Thorough testing before deployment

- Sufficient grey-box and white-box testing
- Fully profiling about system and software

Motivation

Related works

- Many testbeds have been created as supplement to the classical MoteLab (2005, 14 years ago)



Junyan Ma, Jin Wang, Te Zhang. A Survey of Recent Achievements for Wireless Sensor Networks Testbeds, IEEE CyberC 2017

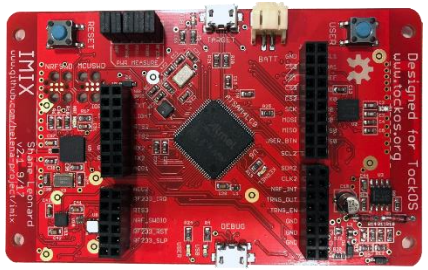
Motivation

Possibility

- Increasing use of ARM Cortex-M3/M4 processors
- Standardized built-in hardware core for debugging
  - ITM & ETM
- General hardware assisted tracing technology
  - beyond GPIO tracing & more internal states
  - continuous tracing VS JTAG

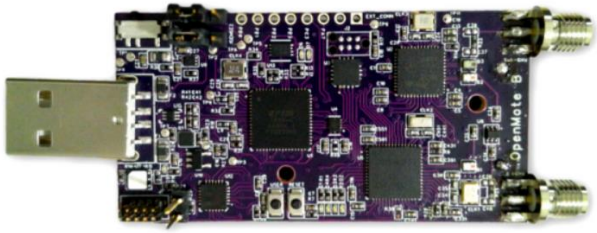
MCU	Architecture	ITM	ETM	Year
STM32F103	Cortex-M3	✔	✔	2012
CC2538	Cortex-M3	✔	✘	2015
EZR32WG	Cortex-M4	✔	✔	2016
SAM4L	Cortex-M4	✔	✔	2017
CC2652	Cortex-M4	✔	✘	2018

Hail



IMIX

Opal



OpenMote

Non-invasive profiling of IoT Devices

Hardware Assisted Tracing

What is Hardware Assisted Tracing?

- Tracing CPU core execution without impact on its performance.
- No external hardware need to be connected.
- No dependence on the operating systems.

X86

ARM

PowerPC

HW Assisted Tracing technology is widely used on modern processors.

Why IoT Devices need non-invasive profiling

Characteristics of IoT Devices

- Real-time processing
- Wireless communication
- Resources Constrained
- Limited exposure of internal states

Profiling Requirement for IoT Devices

- Keep the testing as close to the real world as possible
- Sufficient grey-box and white-box testing to reveal problems before deployment/in-situ

Hardware Assisted Tracing on Cortex-M3/M4

Data Watchpoint and Trace (DWT)

- **Four configurable comparators:** hardware watchpoint, ETM trigger, PC sampler event trigger, data address sampler event trigger
- **DWT counters:** Clock cycles, Sleep cycles, CPI, Interrupt overhead

Instrumentation Trace Macrocell (ITM)

- emits trace information as packets
- Software trace. Software can write directly to ITM registers
- Hardware trace. The DWT generates these packets, and the ITM emits them
- Time stamping. Timestamps are emitted relative to packets. The ITM contains a 21 bit counter to generate the timestamp

Embedded Trace Macrocell (ETM)

- can trace every instruction executed by MCUs (however, it will slowdown the MCU)

One Port

- Trace Serial Wire Output (SWO)

HATBED Design

HATBED Services

Network-Wide Remote debugging

- Non-intrusive tracing
- Global breakpoints
- Networked-wide distributed assertions

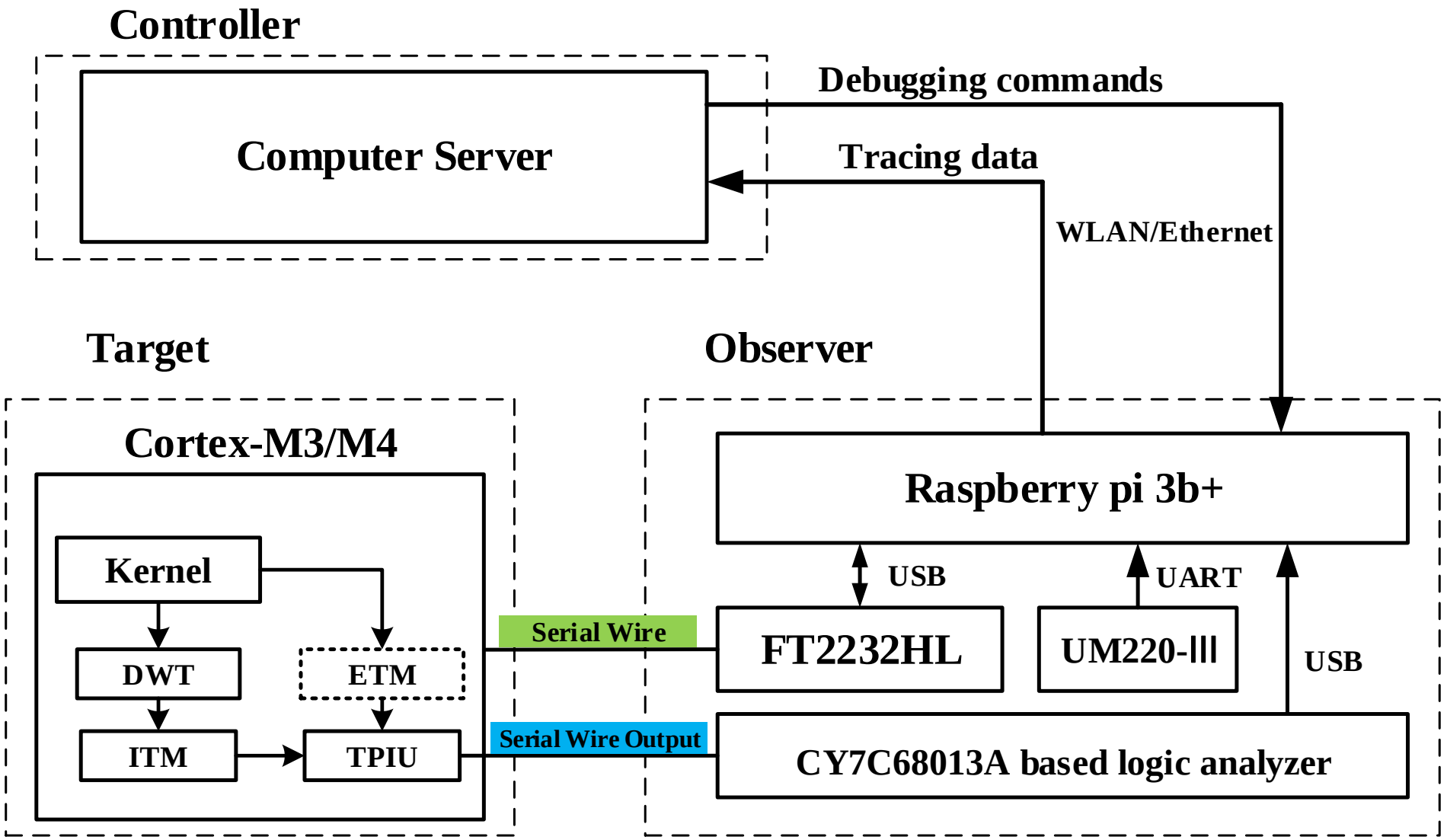
Flexible Software Tracing

- Similar with UART printf
- More flexible and lower latency

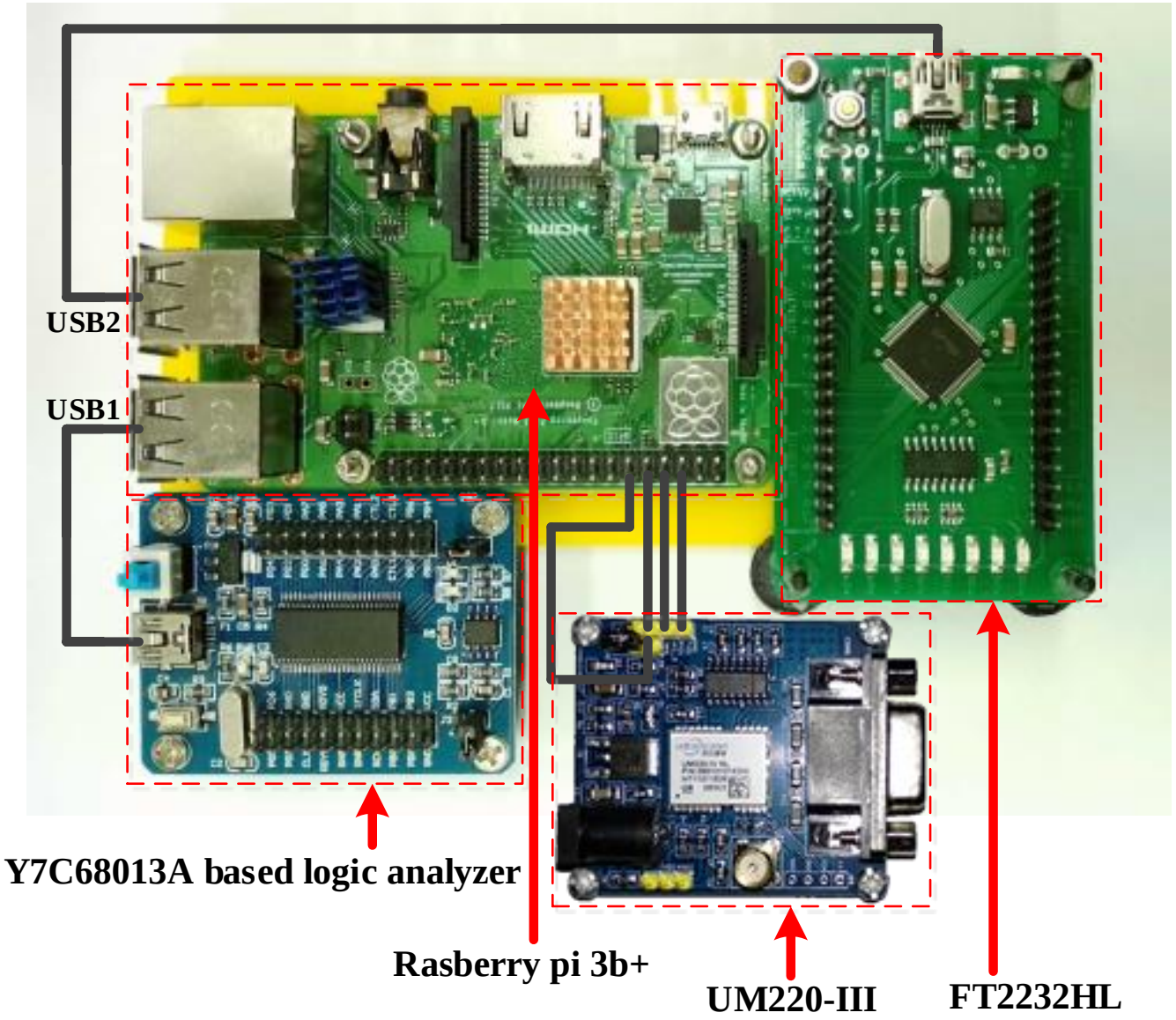
Non-invasive High Precision Software Profiling

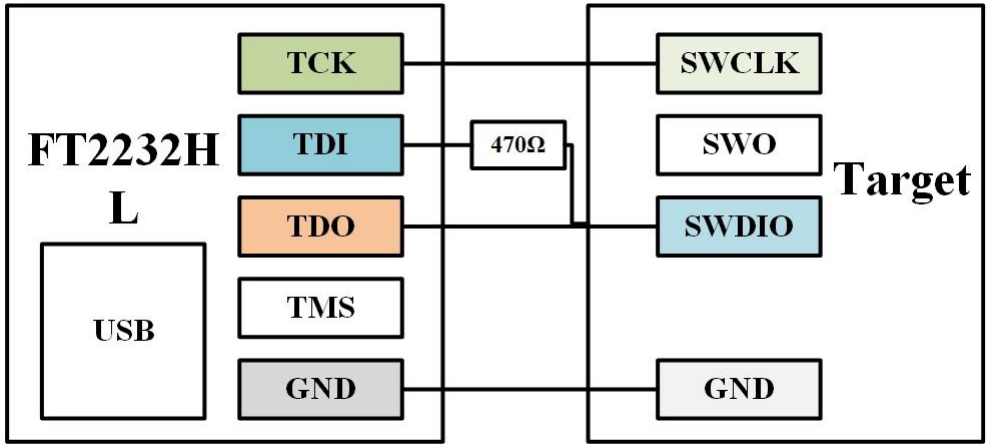
- PC sampling
- Various performance counters
- Hardware watchpoints for capturing changes
- Interrupt events tracing

HATBED Architecture



HATBED Observer





The connect between FT2232HL and Target.

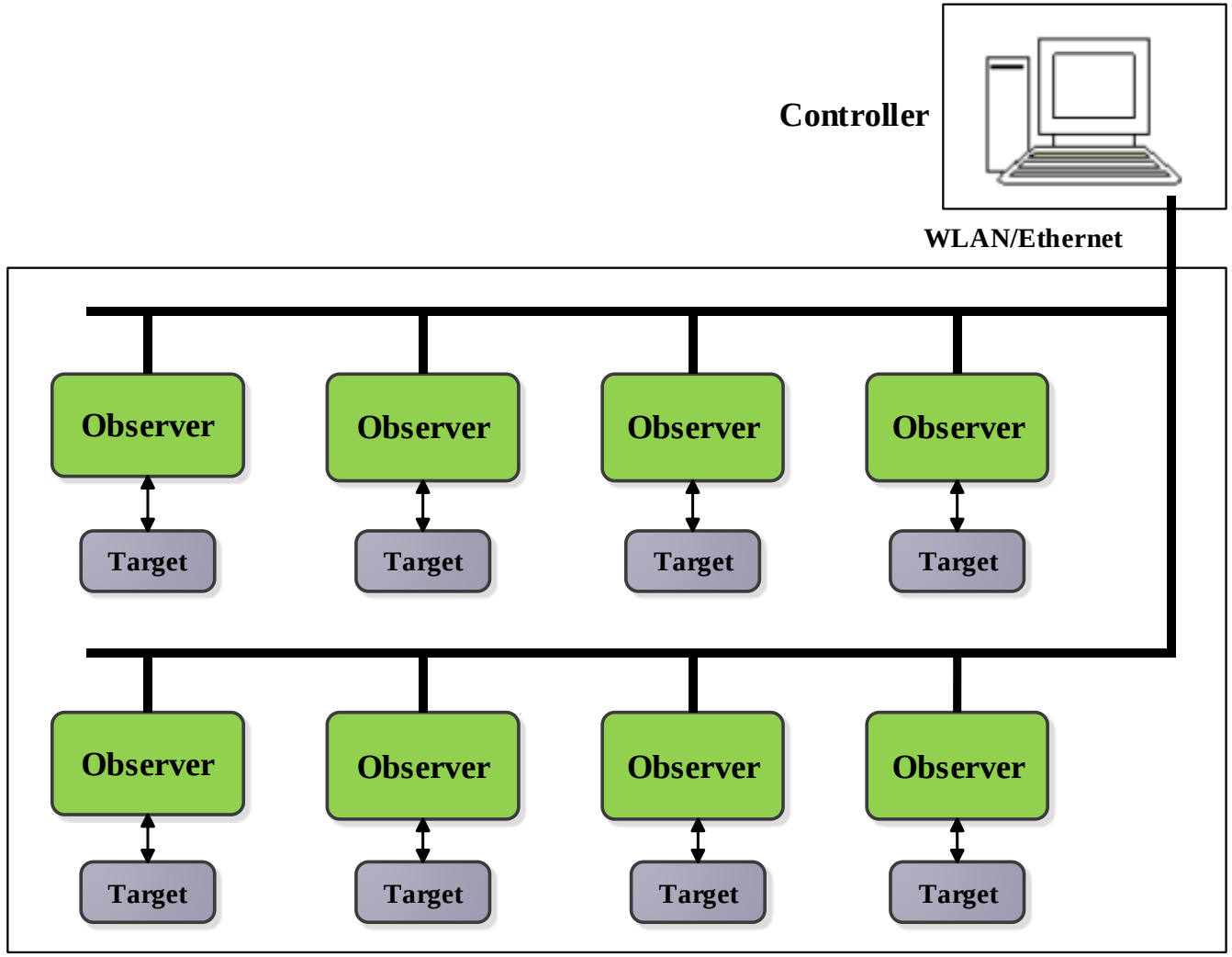
Observer Software Stack

- OpenOCD (Open On-chip debugger)
- GDB of ARM Embedded toolchain (Debugger software)
- Sigrok (Command line logic analyzer)
- Python automated scripts

HATBED Architecture

Testbed Controller

- A powerful computer server.
- Control the whole system
- Data collection and analysis
- Control scripts



The whole system.

Preliminary Evaluation

Overhead of ITM and ETM

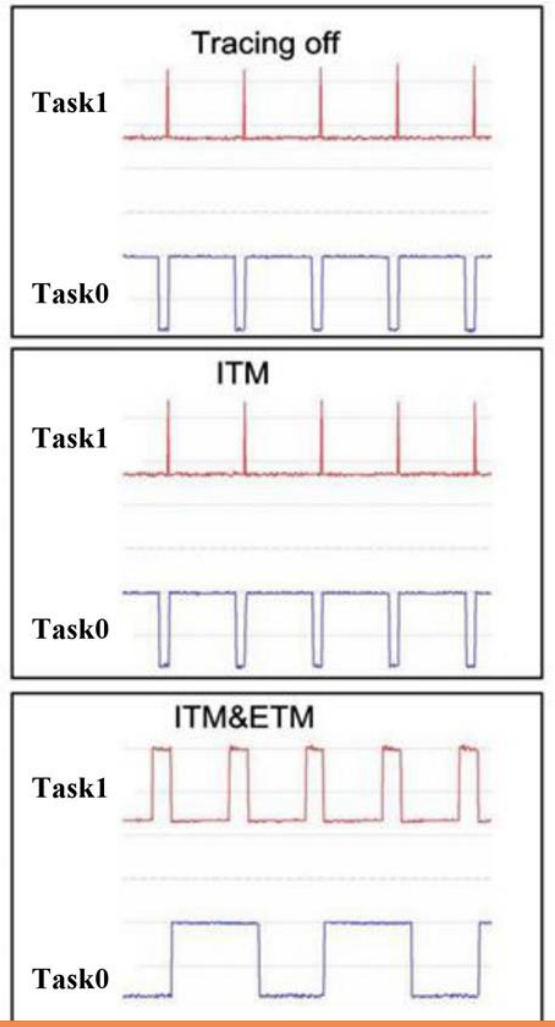
Hardware Settings

- A STM32F103VET6 Cortex-M3 board @ 72MHz
- The baud rate of tracing output set as 8MHz
- Sample rate of logic analyzer set as 24MHz.

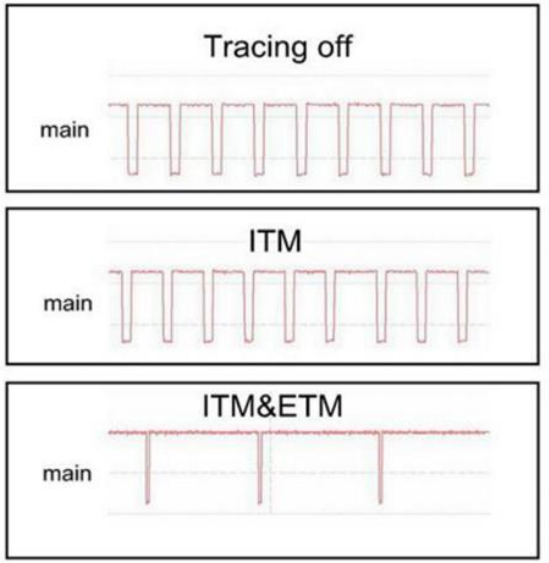
Software Settings

- Bare-metal sorting-based program
- A FreeRTOS-based program
  - Task0(Light LED)
  - Task1(Insert Sort and Bubble Sort)
- Both of them are generated by STM32CubeMX software under Ubuntu14.04 Linux to reduce uncertainty.
- Insert two GPIO flips in the program (~100 ns overhead)
- ITM enable – At all stages
- ETM enable – Trace bubble sort function

FreeRTOS



Bare-metal



GPIO flips are captured by an oscilloscope

ITM enabled – on significant overhead
ETM enabled – Slowdown MCU

Overhead of ITM Print

Sending 4 bytes with ITM_Print only takes 2.5 us

38 bytes are sent by using different methods

- ITM_Print -- A serial debugging method by using ITM output
- UART printf -- Rewrite the printf output to UART port (Using Microlib)
- HAL_UART_Transmit -- Direct register operation

Name	ITM_Print	HAL_UART_Transmit	UART printf
Time(us)	86	1500	3650

Energy Consumption

- Measuring tool – Tektronix PA1000 Power Analyzer
- ITM enable – At all stages
- ETM enable – Trace bubble sort function
- Multiple measurements are averaged

Name	Voltage (V)	Current (mA)	Power(mW)
Tracing OFF	4.841	85.79	415.3
ITM	4.84	87.15	427.1
ITM&ETM	4.845	82.06	397.3

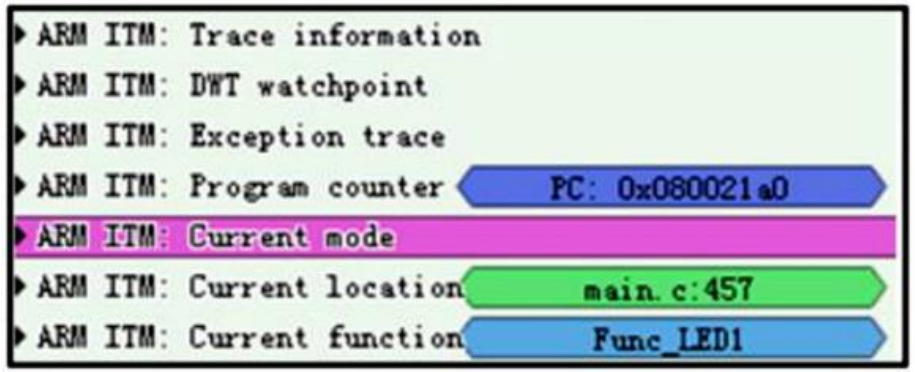
FreeRTOS

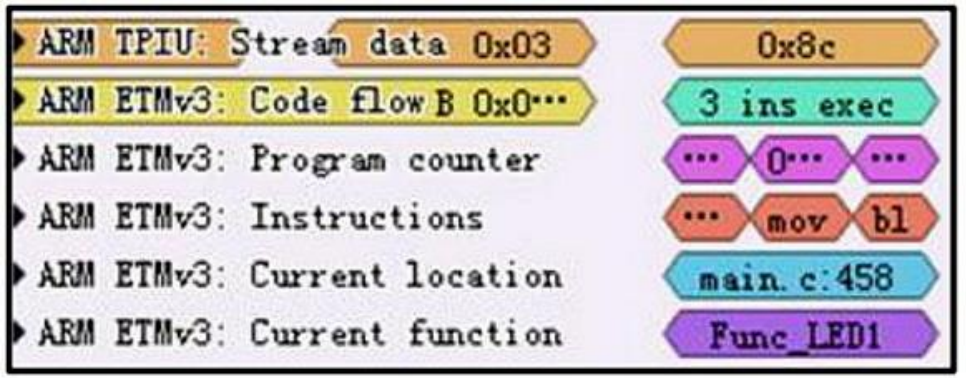
Name	Voltage (V)	Current (mA)	Power(mW)
Tracing OFF	4.837	84.24	407.46
ITM	4.842	85.42	413.6
ITM&ETM	4.853	82.57	400.63

Bare-metal

Coverage with ETM

- Slows down the MCU but records total internal details
- Should be used with caution.





Captured traces include instructions, current location, and current function can help us to analyze code coverage.

Code Coverage with ITM

Global counter + DWT

- insert a global counter into the beginning of functions

- 6 functions are trace by a global counter

PC Sample

- others 4 functions will be left for PC sample

	Name	Global counter value	Capture or Not
Global counter	Task0	0x0A	✓
	Task1	0x0B	✓
	ITM_Print	0x0C	✓
	Bubble_sort	0x0D	✓
	FFT	0x0E	✓
	IFFT	0x0F	✓
PC sample	HAL_GPIOWrite_Pin	null	✗
	osDelay	null	✓
	Binary_InsertSort	null	✓
	Find_InsertIndex	null	✓

We captured 9 of the 10 functions by PC and DWT watchpoint.

Conclusion

Conclusion

HATBED at present

- By using standard components, we design a low-cost (less than $60 per observer) testbed.
- The preliminary evaluation verified the feasibility of our design.
- ITM brings good visibility of internal MCU and incurs little overhead (Time and Energy)
- The feasibility of mechanism for time synchronization between ITM trace and outside are verified
- ETM tracing is not suitable for real-time benchmarking

HATBED in the future

- Testbed with 10 to 20 nodes
- Repeat experiments by Flocklab on HATBED
- More realistic scenario benchmarking to show the advantage of ITM/ETM
- Code coverage estimation by PC sampling

Many thanks for your attention!

Q & A