ARC DESIGN CONTEST

指導老師：張錫嘉 教授
參賽隊員：劉力瑋.宋季轅.郭緯謙

Outline

- Background and Motivation
- Thermal Sensor Overview
- Environment Setting
- Feature Selection
- Data Analysis Method and Result
- Conclusion

Background & Motivation

Background

- **Accuracy**
  - An indicator of thermal sensor
  - How close you are to the actual value
- **Calibration methods**
  - Dual sensing variation cancelling

Background

- **Thermal sensor output**
  - Worse linearity (Overly ideal simulation process)
  - Accuracy Issue (Time consuming simulation)

Motivation

- **Our goal**
  - Propose an accurate calibration method based on thermal sensor
  - Build a machine learning model to achieve the goal on EMSK

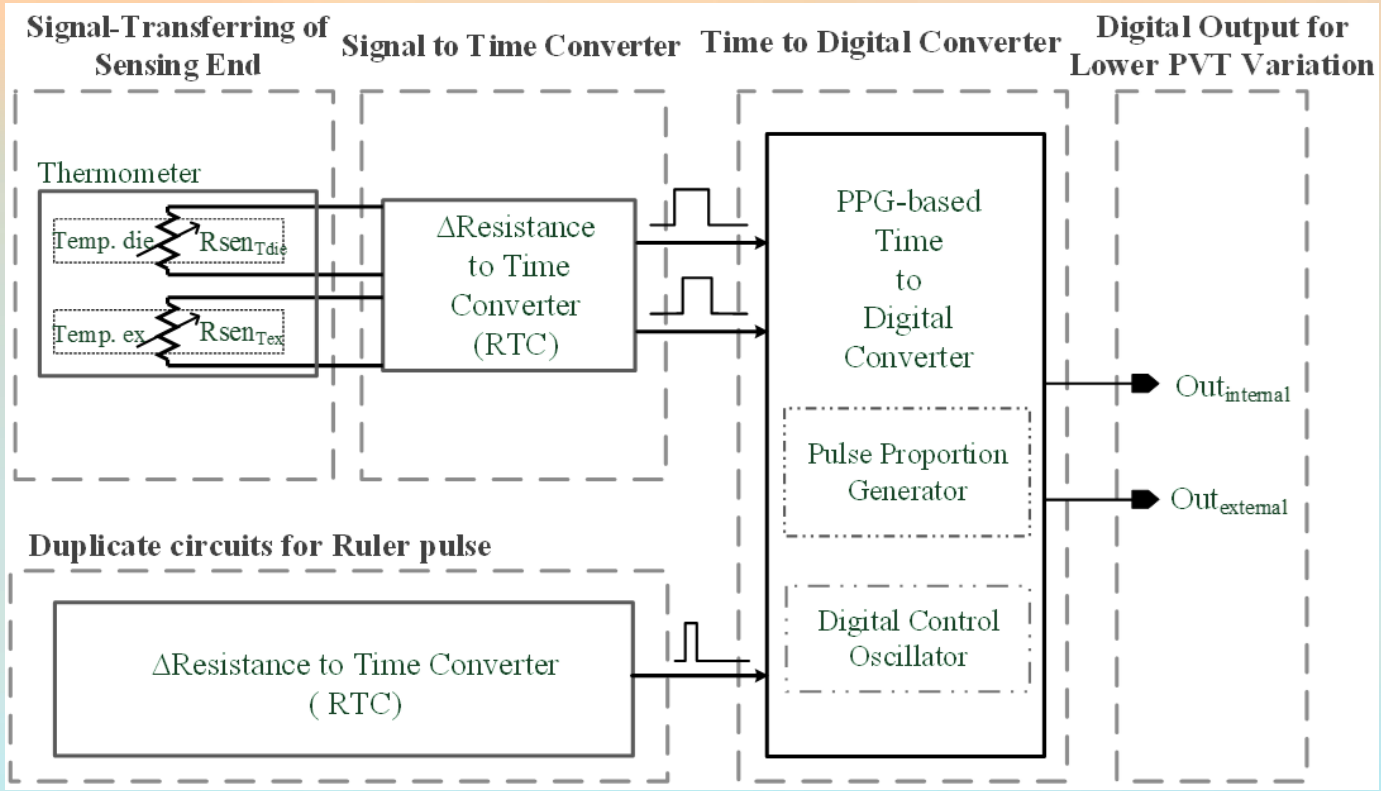
Thermal Sensor Overview

Thermal Sensor Overview

- **DIP 32 Package**
- **Technology Node: 0.18um**
- **IO Pad Voltage: 1.8V**
- **Thermal Sensor**
  - Temperature range: 0~100℃
- **Calibration Features**
  - Dual sensing variation cancelling
  - Pulse Proportion Generator (PPG) for reducing PVT variations

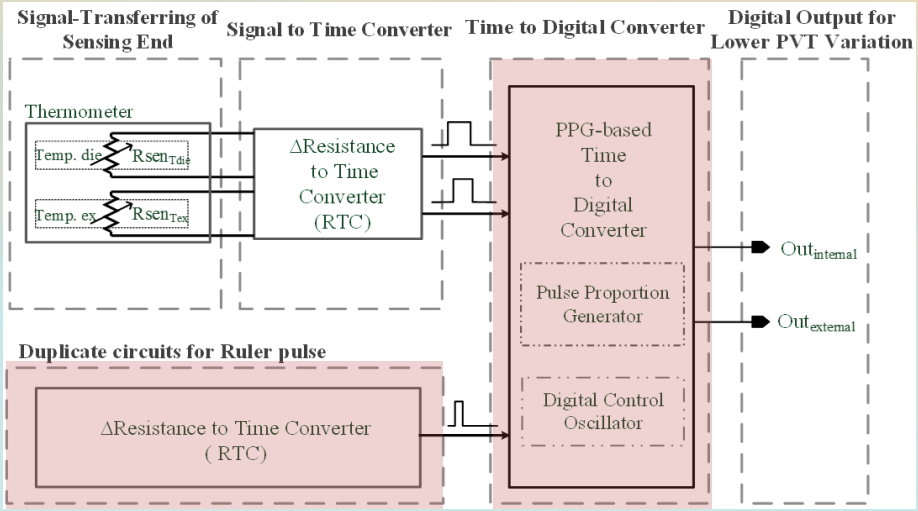
Thermal Sensor Overview

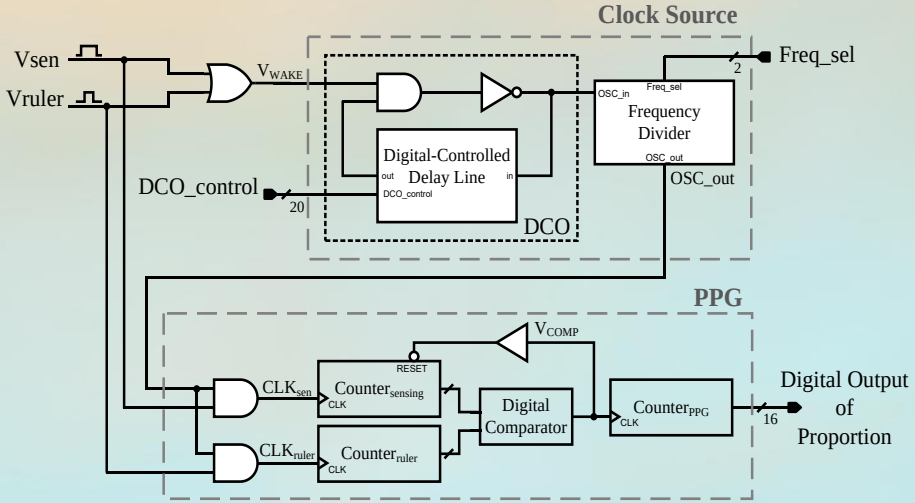
Dual Sensing Variation cancelling



Thermal Sensor Overview

- Pulse Proportion Generator
  - Generate a ruler time-pulse as reference
  - Taking proportion between V_{ruler} and V_{sen}
  - To compensate the P-V-T variations



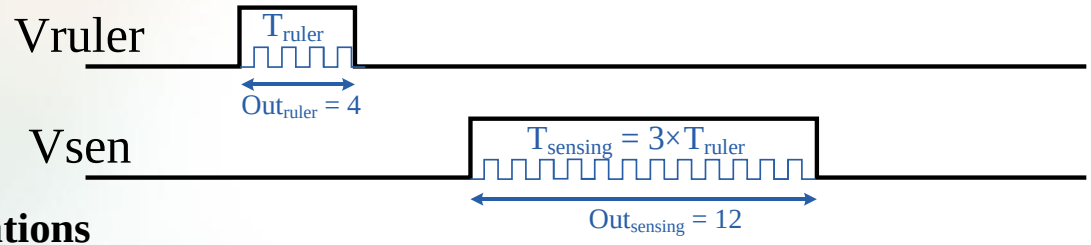


Thermal Sensor Overview

- PVT variations will be eliminated

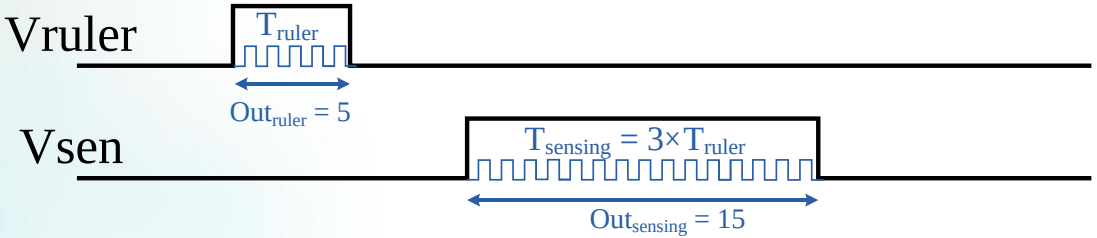
Without variations

$$P_{ideal} = \frac{k \times Sense}{k \times Ruler}$$



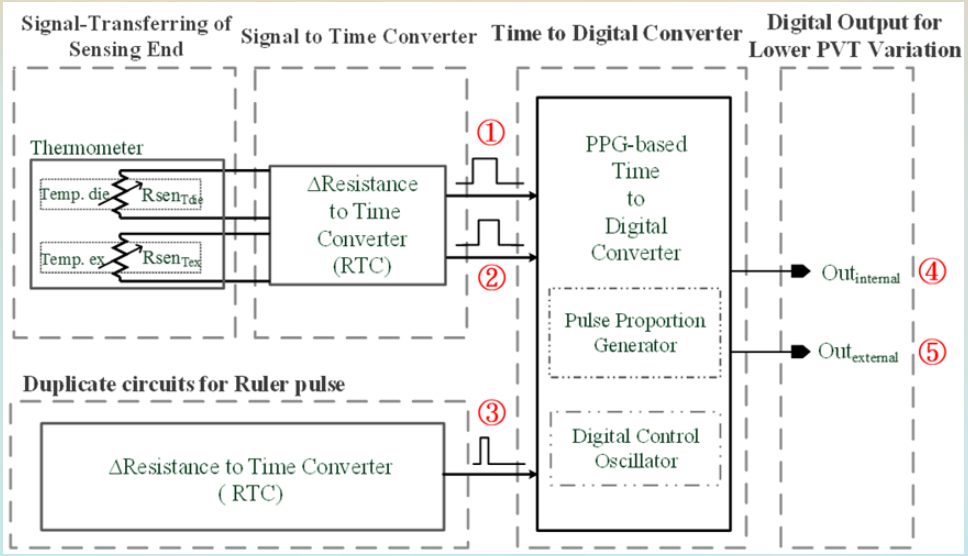
With variations

- Supply voltage
- Temperature



Thermal Sensor Overview

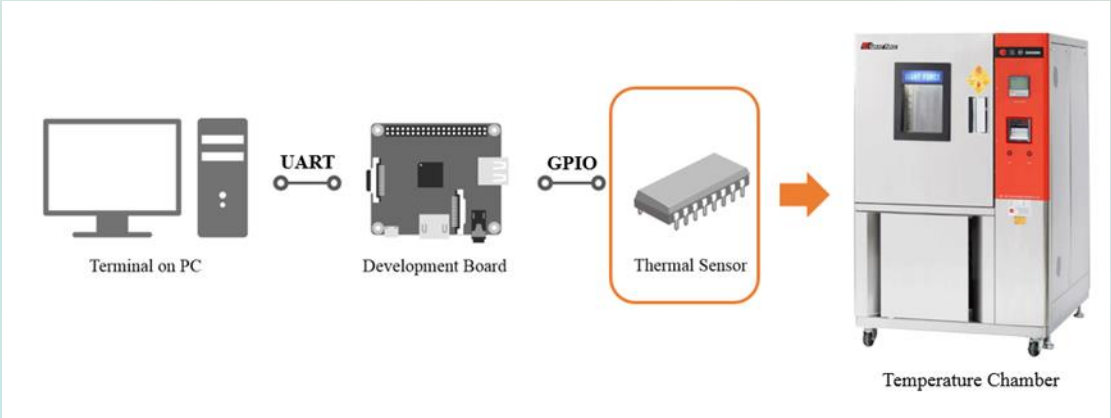
- **Thermal Sensor Output**
  - ①Sense_in ②Sense_ex ③Ruler
  - ④PPG_in=Sense_in/Ruler ⑤PPG_ex=Sense_ex/Ruler



Environment Setting

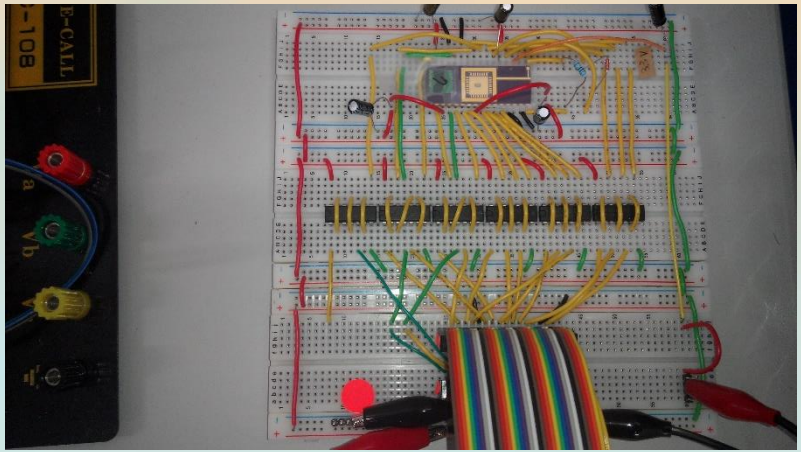
Environment Setting

- **System Architecture**
  - Embedded ARC Starter Kit
  - Thermal Sensor Circuit
  - UART/GPIO
  - Temperature chamber

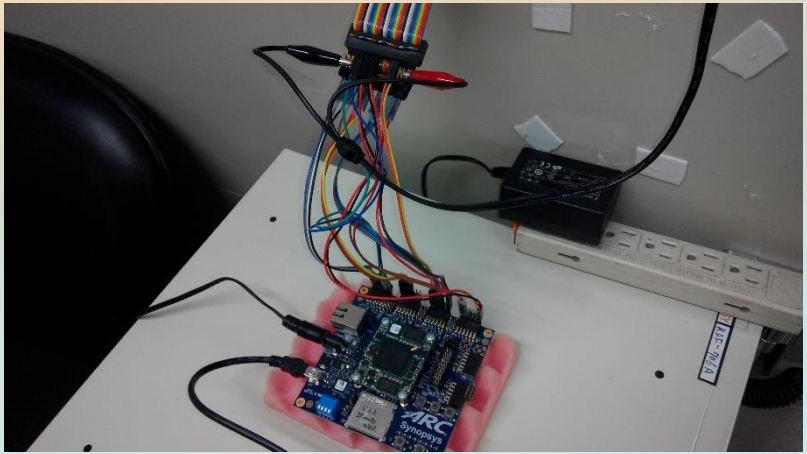


Environment Setting

Thermal Sensor Circuit



ARC EMSK Platform



Environment Setting

- **PMOD**
  - Set PMOD as GPIO to control our signal

```
port_output = gpio_get_dev(DW_GPIO_PORT_C);
port_output -> gpio_open(0x00ffff00);
port_output ->gpio_control(GPIO_CMD_SET_BIT_DIR_OUTPUT, (void *)0x00ffff00);
port_output ->gpio_control(GPIO_CMD_DIS_BIT_INT, (void *)0x00ffff00);
```

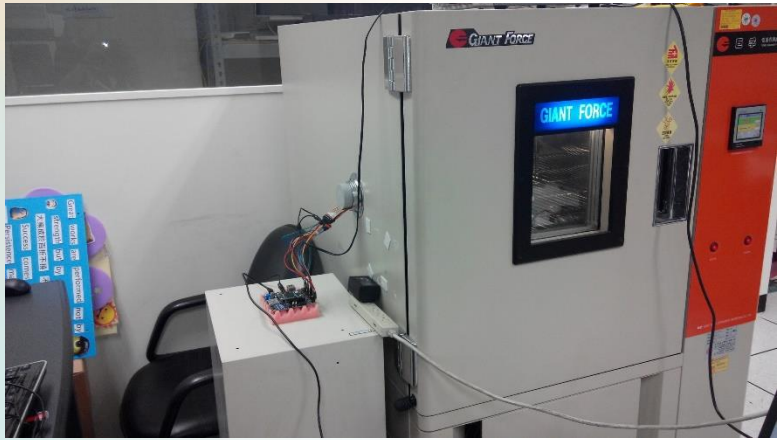
```
port_output-> gpio_write(PATTERN_EXCEL[count]<<PMOD_OFFSET0, MASK_GPIO0);
```

- **Timer interrupt**
  - At specific frequency range, we output signal to IC from our specific pattern

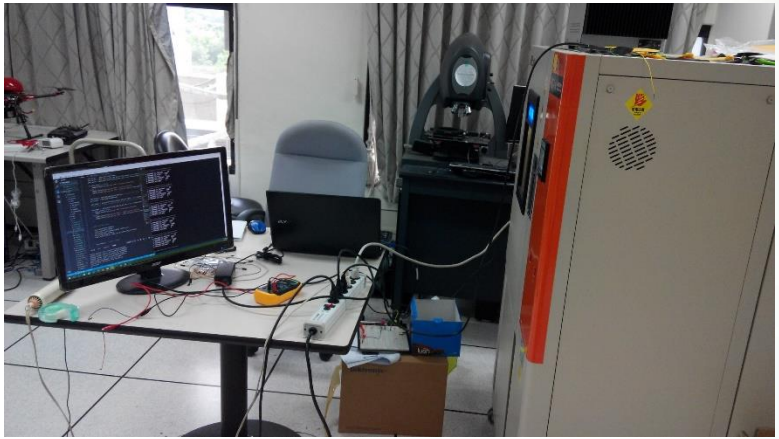
```
int_disable(INTNO_TIMER0);
timer_stop(INTNO_TIMER0);// stop timer for setting handler
int_handler_install(INTNO_TIMER0, timer0_isr);
int_enable(INTNO_TIMER0);
timer_start(TIMER_0, TIMER_CTRL_IE, BOARD_CPU_CLOCK/1000);
```

Environment Setting

Put the Circuit In Chamber



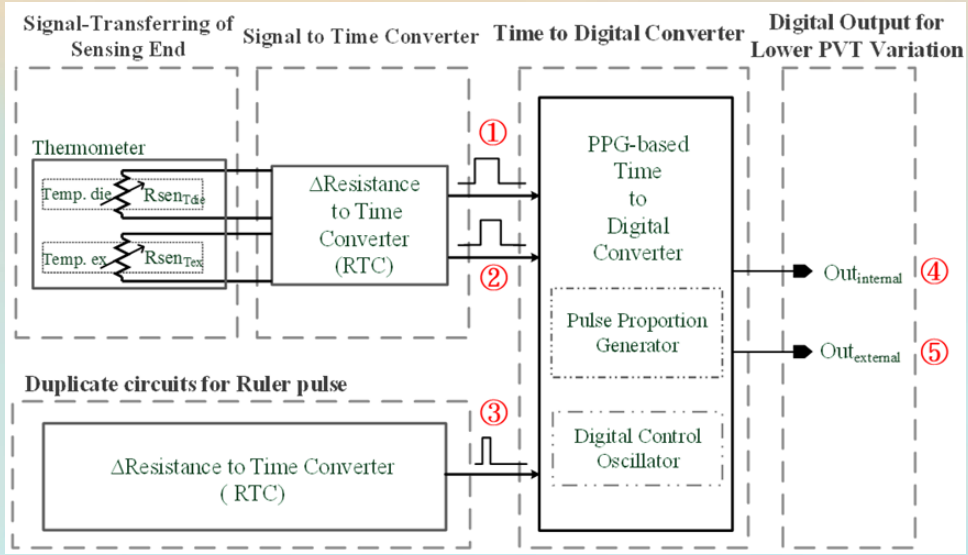
Start Data Collection



Feature Selection

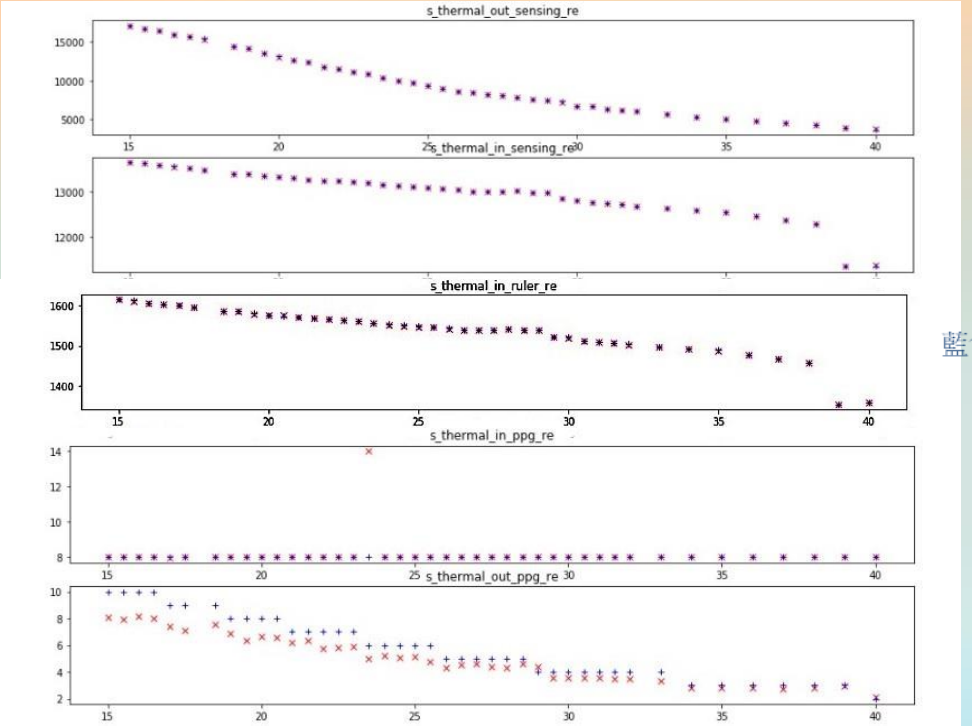
Feature Selection

- **Five Output**
  - ①Sense_in ②Sense_ex ③Ruler
  - ④PPG_in=Sense_in/Ruler ⑤PPG_ex=Sense_ex/Ruler
- **Feature Selection**
  - Linearity
  - Function



Feature Selection - Linearity

- **Ruler breaks the linearity of sensing output**



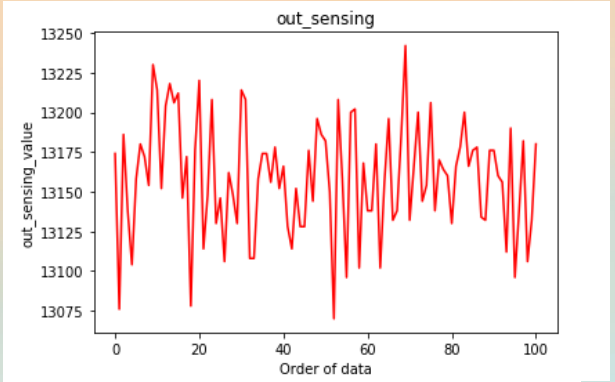
藍色:median 紅色:mean

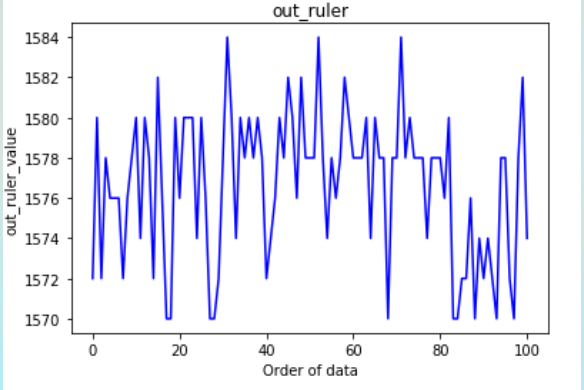
Feature Selection – PPG Function

- **The difference between SENSE_in and Ruler is not proportional, same as SENSE_ex and Ruler**

$$P_{ideal} = \frac{k \times Sense}{k \times Ruler}$$

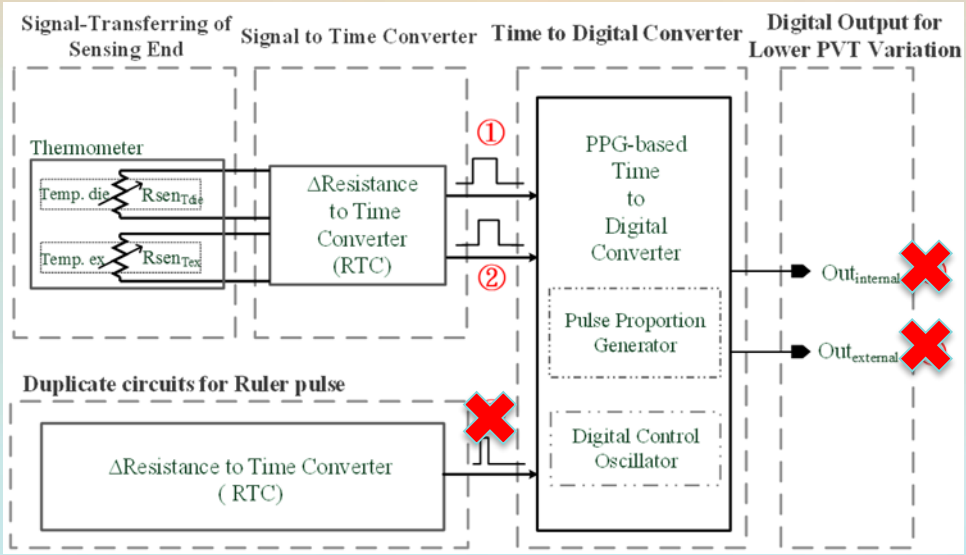
$$P_{measurement} = \frac{k \times Sense + n}{k \times Ruler + n'}$$





Feature Selection

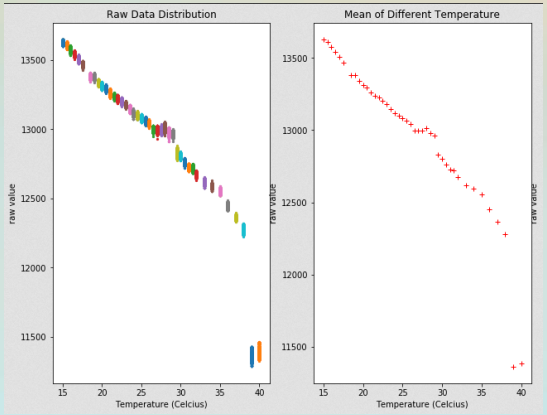
- **We only chose Sense_in and Sense_ex as training input**
  - Ruler does not have great linearity in small range
  - The variation between ruler and sensing is not constant in 15℃ to 40℃



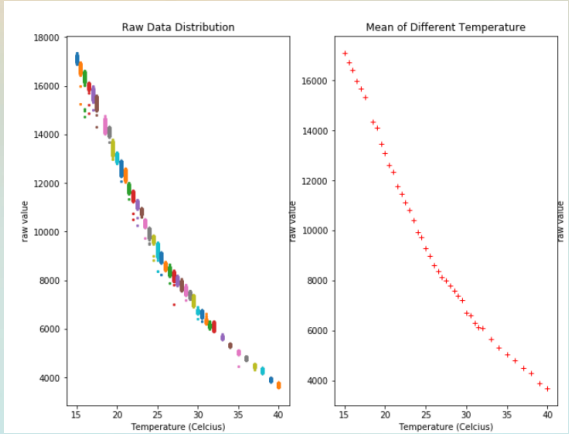
Data Analysis &
Machine Learning Based Method

Data Analysis

- **Sense_out is the linear data**
- **Sense_in can be the auxiliary data**



(s_thermal_in_sensing)

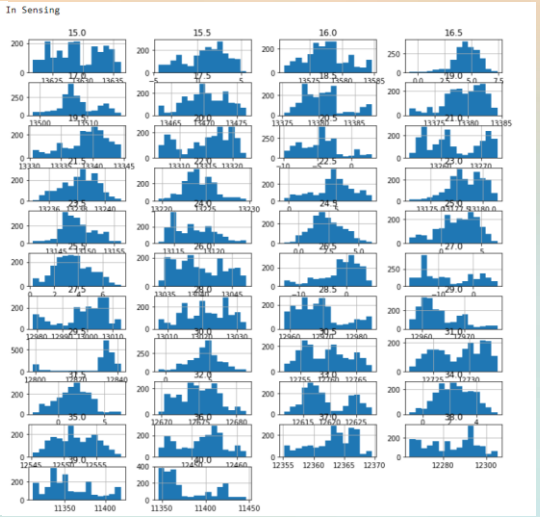


(s_thermal_out_sensing)

Data Analysis

- **Data is not Gaussian distribution**
  - **Use moving average to smooth raw data**

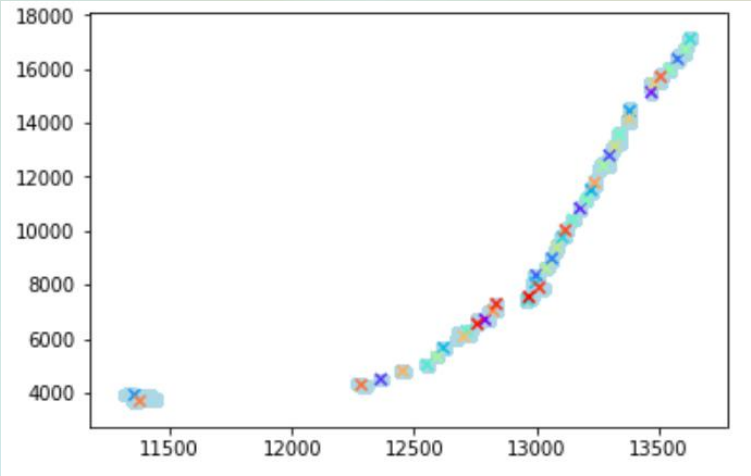
(thermal_out_sensing)



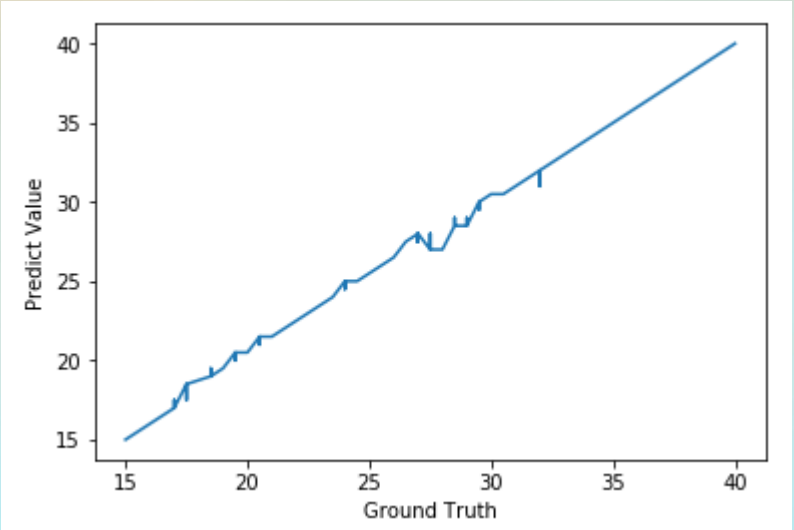
(thermal_in_sensing)

K-mean Clustering

- **X means clustering center**
  - Centers are very close to each other
- **Result**
  - Only 87% accuracy falls on ±0.5℃



(x: in_sensing y:out_sensing)



Multiple Linear Regression

- **Equation:**

$$\beta_0 out^3 + \beta_1 out^2 + \beta_2 out^1 + \beta_3 in^2 + \beta_4 in^1 + \beta_5 = \text{Estimated Temperature}$$

- **Using minimum mean square error to train weights**

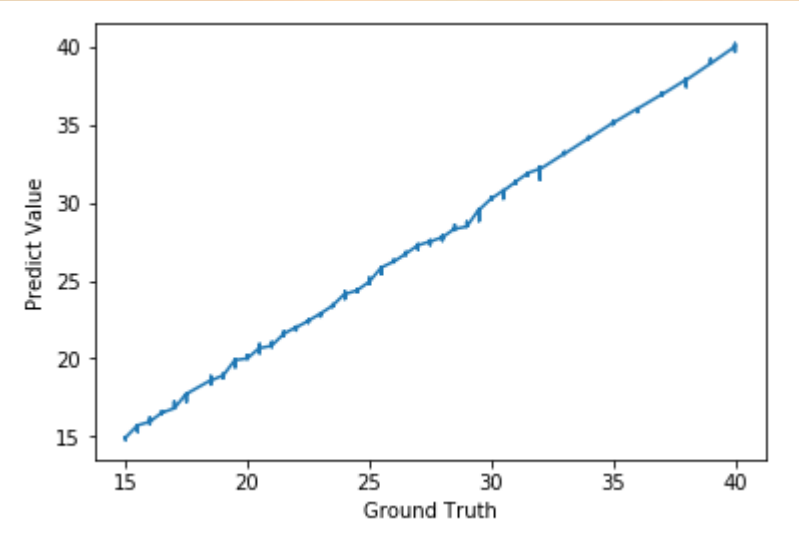
$$L = \sum_{i=1}^{n} \varepsilon_i^2 = \varepsilon^{'}\varepsilon = (y - X\beta)^{'}(y - X\beta)$$

$$\frac{\partial L}{\partial \beta} = 0 \Rightarrow X^{'}X\hat{\beta} = X^{'}y$$

L: square error
y: true temperature
β: weight
X:input data matrix

Multiple Linear Regression

- **Result**
  - 97.48% accuracy falls on ±0.5℃



Demo Video

Demo Video(open the link at picture)

Conclusion

Conclusion

- **An accurate calibration method is proposed**
  - multiple linear regression
- **Improve the accuracy from 1℃ to 0.5℃**

Thank you