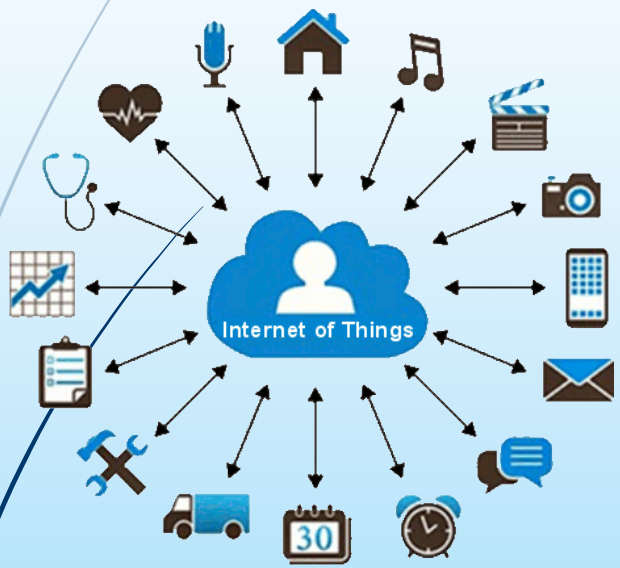
Embedded Systems - API Classes



- API
  - DigitalIn, DigitalOut
  - BusIn, BusOut
  - AnalogIn, AnalogOut
  - Interrupt
  - Timer, TimeOut, Ticker
  - PWM
  - Serial
  - I2C

mbed GPIO: DigitalIn

- 1-bit digital input
- https://os.mbed.com/docs/mbed-os/v6.15/apis/digitalin.html
- Main class public member functions

```
    DigitalIn(PinName pin)
    DigitalIn(PinName pin, PinMode mode)
Void mode(PinMode mode) // mode: PullUp, PullDown, PullNone
 int read()
    operator int()
```

mbed GPIO: DigitalOut

- 1-bit digital output
- https://os.mbed.com/docs/mbed-os/v6.15/apis/digitalout.html
- Main class public member functions

	DigitalOut(PinName pin)
	DigitalOut(PinName pin, int value)
void	write(int value)
int	read()
DigitalOut&	operator=(int value)
	operator int()

mbed GPIO: DigitalIn/DigitalOut Example

```
#include "mbed.h"

DigitalIn  mypin(SW2, PullUp);
DigitalOut myled(LED_A);

void main(void) {
    long long i = 0;
    long long j = 0;

    mypin.mode(PullDown);

    while(1) {
        i++;
        j += (mypin == 1) ? 1 : -1;
        myled = mypin;
        myled.write(mypin.read());
    }
}
```

mbed GPIO: BusIn

- A set of digital inputs (up to16 bits)
- https://os.mbed.com/docs/mbed-os/v6.15/apis/busin.html

	BusIn(PinName lsb, PinName p1=NC, PinName p2=NC, ..., PinName p15=NC)
void	mode(PinMode mode)// PullUp, PullDown, PullNone
int	read()
	operator int()
DigitalIn&	operator[](int index)

mbed GPIO: BusOut

- A set of digital outputs (up to16 bits)
- https://os.mbed.com/docs/mbed-os/v6.15/apis/busout.html

	`BusOut(PinName lsb, PinName p1=NC, PinName p2=NC, ..., PinName p15=NC)`
`void`	`write(int value)`
`int`	`read()`
`BusOut&`	`operator=(int value)`
	`operator int()`
`DigitalOut&`	`operator[](int index)`

mbed GPIO: Bus examples

```
#include "mbed.h"

BusOut      myleds(LED_A, LED_B, LED_C, LED_C);
DigitalIn   mysw;

void main(void) {
    mysw.mode(PullUp);

    while(1) {
        if(!mysw) myleds = myleds + 1;
        // if (!mysw.read()) myleds.write(myleds.read() + 1);
    }
}
```

mbed GPIO: Practical Example

Simple Exercise 1:

- Modify the blinky example to create a BUS with the available LEDs and utilize the USER_BUTTON to ‘increment’ the displayed value (light ON = 1; light OFF = 0)

mbed ADC: AnalogIn

- A/D Converter
- https://os.mbed.com/docs/mbed-os/v6.15/apis/analogin.html

	`AnalogIn(PinName pin)`
`float`	`read()` `// 0.0 <= return value <= 1.0`
`unsigned short`	`read_u16()` `// 0 <= return value <= 65535`
	`operator float()`

mbed DAC: AnalogOut

- D/A Converter
- https://os.mbed.com/docs/mbed-os/v6.15/apis/analogout.html

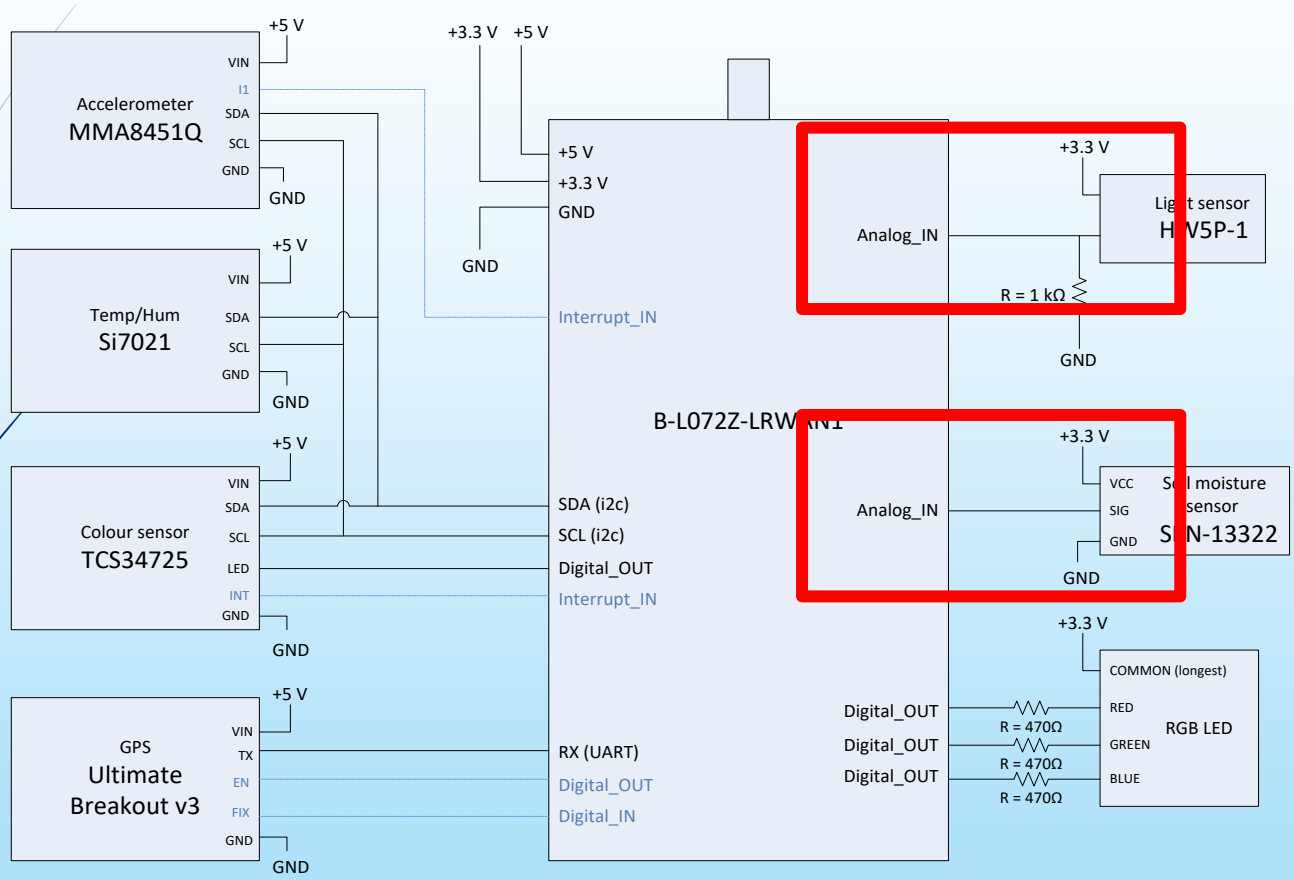
	`AnalogOut(PinName pin)`
`void`	`write(float value)` `// 0.0 <= value <= 1.0`
`void`	`write_u16(unsigned short value)` `// 0 <= value <= 65535`
`float`	`read()` `// 0.0 <=, <= 1.0`
`AnalogOut&`	`operator=(float value)`
	`operator float()`

ThisThread::sleep_for(1);

mbed ADC: AnalogIn Examples

```
#include "mbed.h"

AnalogIn input(A0);

void main(void) {
    uint16_t samples[1024];

    for (int i=0. i<1024; i++) {
        samples[i] = input.read_u16();
        ThisThread::sleep_for(1);// sampling period
    }
}
```

Block Diagram (Recall)



… Analog_IN inputs

mbed ADC: AnalogIn Practical Example

Simple Exercise 2:

- Input a voltage to the platform with two resistors (or a resistor and a LED), sample it and show the value (in volts)

- Output a sinewave voltage from the platform with a frequency of 1Hz and 100 samples/period. Printf the output values

  - Hint: See example here:

`https://os.mbed.com/docs/mbed-os/v6.15/apis/analogout.html`

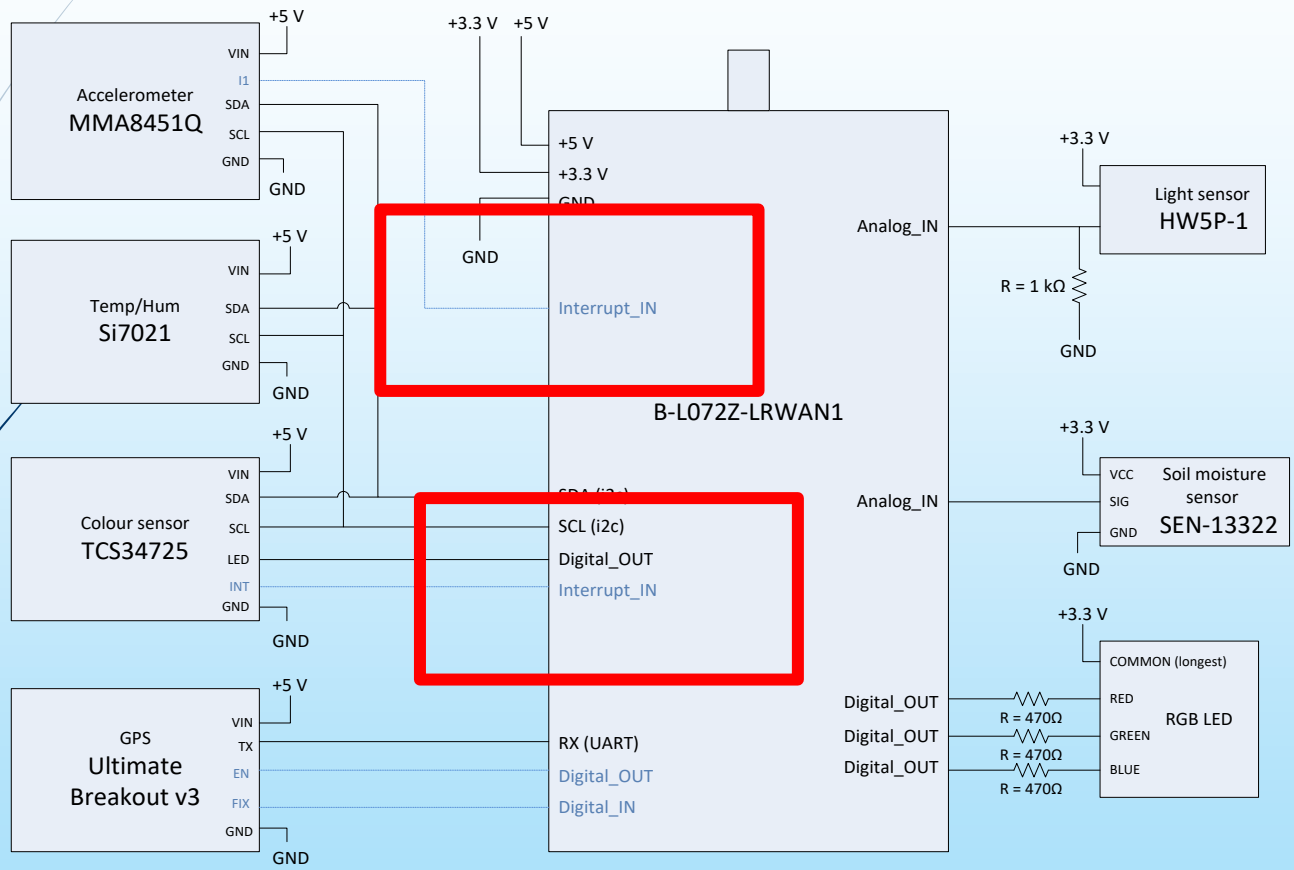
mbed IRQ: `InterruptIn`

- Triggers a routine and interrupts the main thread when a digital input pin changes (*event*)
- `https://os.mbed.com/docs/mbed-os/v6.15/apis/interruptin.html`

```
InterruptIn(PinName pin)
Void mode(PinMode mode) // mode: PullUp, PullDown, PullNone
 int read()
operator int()
void rise(void (*handler)(void))
void fall(void (*handler)(void))
```

- Use `rise(NULL)`/`fall(NULL)` to disable the interrupt

Block Diagram (Recall)



… Interrupts coming from two sensors to the platform

mbed IRQ: InterruptIn Example

```
#include "mbed.h"
InterruptIn mypin(p28);
BusOut      myleds(LED_A, LED_B, LED_C, LED_D);

void mypin_handler(void) {myleds = myleds + 1;}

void main(void) {
    mypin.mode(PullUp);
    mypin.fall(mypin_handler); // use mypin.fall(NULL) to disable

    myleds = 0;

    while(true) { // umm …
    }
}
```

mbed IRQ: InterruptIn Another Example

```
#include "mbed.h"
InterruptIn mypin(p28);
BusOut      myleds(LED_A, LED_B, LED_C, LED_D);
bool        count = false;
void mypin_handler(void) {count = true;} //ISR

void main(void) {
    mypin.mode(PullUp);
    mypin.fall(mypin_handler);   // use mypin.fall(NULL) to disable
    myleds = 0;

    while(true) {
        if(count) {
            count = false;
            myleds = myleds + 1;  // no += operator for BusOut
        }
    }
}
```

mbed IRQ: `InterruptIn` Practical Example

Simple Exercise 3:

- Modify the blinky example to create a BUS with the available LEDs and assign the USER_BUTTON to an interrupt-generating event whose service routine ‘increments’ the displayed value (light ON = 1; light OFF = 0)

mbed Timers: Timer

- Time up to máx. $2^{31} - 1$ µs (~ 35 minutos)
- https://os.mbed.com/docs/mbed-os/v6.15/apis/timer.html

Timer
void start()
void stop()
void reset()
float read()
int read_ms()
int read_us()
std::chrono::microseconds elapsed_time()
operator float()

mbed Timers: `Timer` Example

```
#include "mbed.h"

Timer t;

void main(void) { // how much time?
  int   duration_ms;
  float duration_s;

  while(true) {
    t.reset();
    t.start();
    do_something();
    t.stop();
    duration_ms = t.read_ms();
    duration_s = t;
  }
}
```

mbed Timers: `Timeout`

- Set up an interrupt to call a function after a specified delay (máx. 35 minutes)
- `https://os.mbed.com/docs/mbed-os/v6.15/apis/timeout.html`

Timeout
`void attach(void (*handler)(void), float s)`
`void attach_us(void (*handler)(void), int us)`
`void detach()`

- Useful as a ***non-recurring*** count-down

mbed Timers: Timeout Example

```
#include "mbed.h"

DigitalOut          led(LED1);
Timeout             to;

void to_isr(void) {
  led = !led;
}

int main(void) {

  to.attach_us(to_isr, 500000);

  while(true) { // umm …
  }
}
```

mbed Timers: Timeout Another Example

```
#include "mbed.h"

DigitalOut          led(LED1);
Timeout             to;
bool                toggle = false;

void to_isr(void) {toggle = true;}

void main(void) {

  to.attach_us(to_isr, 500000);
  while(true) {

    if(toggle) {
      toggle = false;
      led = !led;
    }
  }
}
```

mbed Timers: `Ticker`

- Sets up a ***recurring*** interrupt (max. period 35 minutes)
- `https://os.mbed.com/docs/mbed-os/v6.15/apis/ticker.html`

Ticker
`void attach(void (*handler)(void), float s)`
`void attach_us(void (*handler)(void), int us)`
`void detach()`

mbed Timers: `Ticker` Example

```
#include "mbed.h"

DigitalOut          led(LED1);
Ticker              tick;
bool                toggle = false;

void tick_isr(void) {toggle = true;}

void main(void) {
  tick.attach_us(tick_isr, 500000);

  while(true) {
    if(toggle) {
      toggle = false;
      led = !led;
    }
  }
}
```

What happens here?

```
#include "mbed.h"

DigitalOut   led(LED_A);
InterruptIn  sw(p10);

Timer        tmr;
Timeout      to;
bool         on = false;
bool         off = false;
bool         timedout = false;

void sw_on_isr(void) {on = true;}
void sw_off_isr(void) {off = true;}
void to_isr(void) {timedout = true;}

void main(void) {
  sw.mode(PullUp);
  sw.fall(sw_on_isr);
  sw.rise(sw_off_isr);
  led = 0;
  while(true) {
    if(on) {
      on = false;
      tmr.reset();
      tmr.start();
    }
    if(off) {
      off = false;
      tmr.stop();
      to.attach_us(to_isr,
        tmr.read_us());
      led = 1;
    }
    if(timedout) {
      timedout = false;
      led = 0;
    }
  }
}
```

mbed Timers: `wait()`

- Delays

```
void  wait(float s)
void  wait_ms(int ms)
void  wait_us(int us)
```

- Disable interrupts => Not to be used in ISR!

- Note: The function wait is deprecated in favor of explicit sleep functions. To sleep, replace wait with ThisThread::sleep_for(). To wait (without sleeping), call wait_us.

Mbed: Practical Example

Simple Exercise 4:

- Replicate the previous code (slide 26) in your board. Use the elapse_time() method instead of the read_us() one.
  - Hint:

    https://os.mbed.com/docs/mbed-os/v6.15/apis/timer.html

mbed PWM: PwmOut

- Pulse Width Modulation output
- https://os.mbed.com/docs/mbed-os/v6.15/apis/pwmout.html

```
        PwmOut(PinName pin)
void    period(float s)
void    period_ms(int ms)
void    period_us(int us)
void    pulsewidth(float s)
void    pulsewidth_ms(int ms)
void    pulsewidth_us(int us)
void    write(float duty_cycle)
        operator=(float duty_cycle)
float   read() // returns the duty cycle (0.0 <=, <= 1.0)
        operator float()
```

mbed PWM: PwmOut Example

```
#include "mbed.h"
PwmOut    led(LED1);
Ticker    tick;
bool      update;

void tick_isr(void) {update = true;}

void main(void) {
  int     width_us = 0;

  led.period_us(1000);
  led.pulsewidth(width_us);

  tick.attach_us(tick_isr, 500);

  while(true) {
    if(update) {
      width_us += width_us < 1000 ? 1 : -1000;
      led.pulsewidth_us(width_us);
      update = false;
    }
  }
}
```

Mbed: Practical Example

Simple Exercise 5:

- Replicate the previous code (slide 30) in your board.

mbed UART: Serial

- Serial port (simplified)
  - https://os.mbed.com/docs/mbed-os/v5.15/apis/serial.html
- BufferedSerial described here!
  - https://os.mbed.com/docs/mbed-os/v6.15/apis/serial-uart-apis.html

```
Serial(PinName tx, PinName rx)
void baud(int baudrate)
void format(int data_bits, Parity parity, int stop_bits)
                    // parity: Forced0, Forced1, None, Odd, Even
 int readable()     // true when a char is received
 int writeable()    // true if a char can be accepted for tx
operator int()
void attach(void (*handler)(void), IrqType type)
                    // type: RxIrq, TxIrq

                    // Beware that the Serial RxIrq IRQ is not cleared
                    // until the char is read by the processor
```

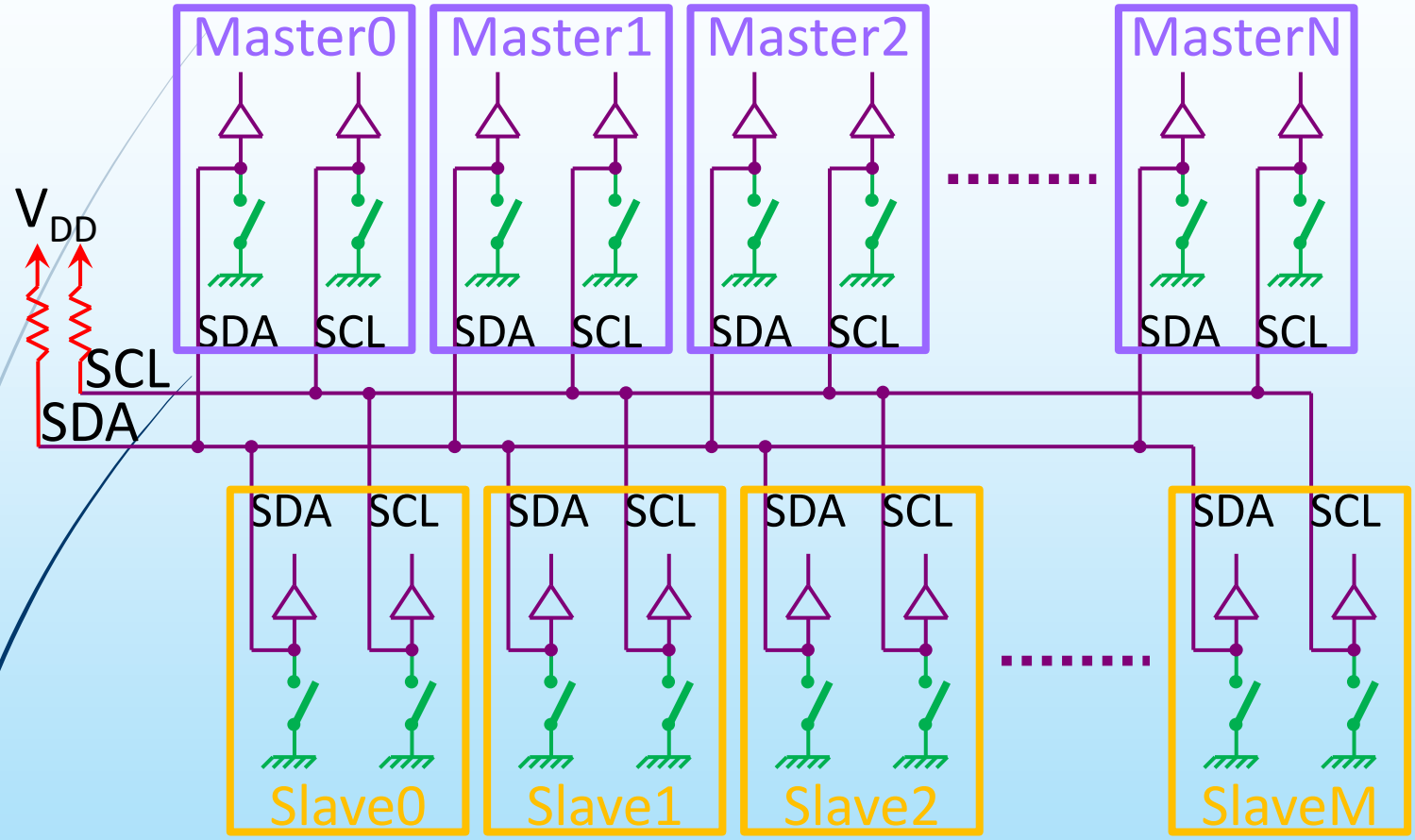
mbed UART: `Serial`

- `Serial` inherits from class `Stream`
- The following methods are also available:
  - `putc(char)` send a character to the UART
  - `char getc()` receive a character from the UART
  - `printf(...)` send a formated chain to the UART

mbed UART: Example

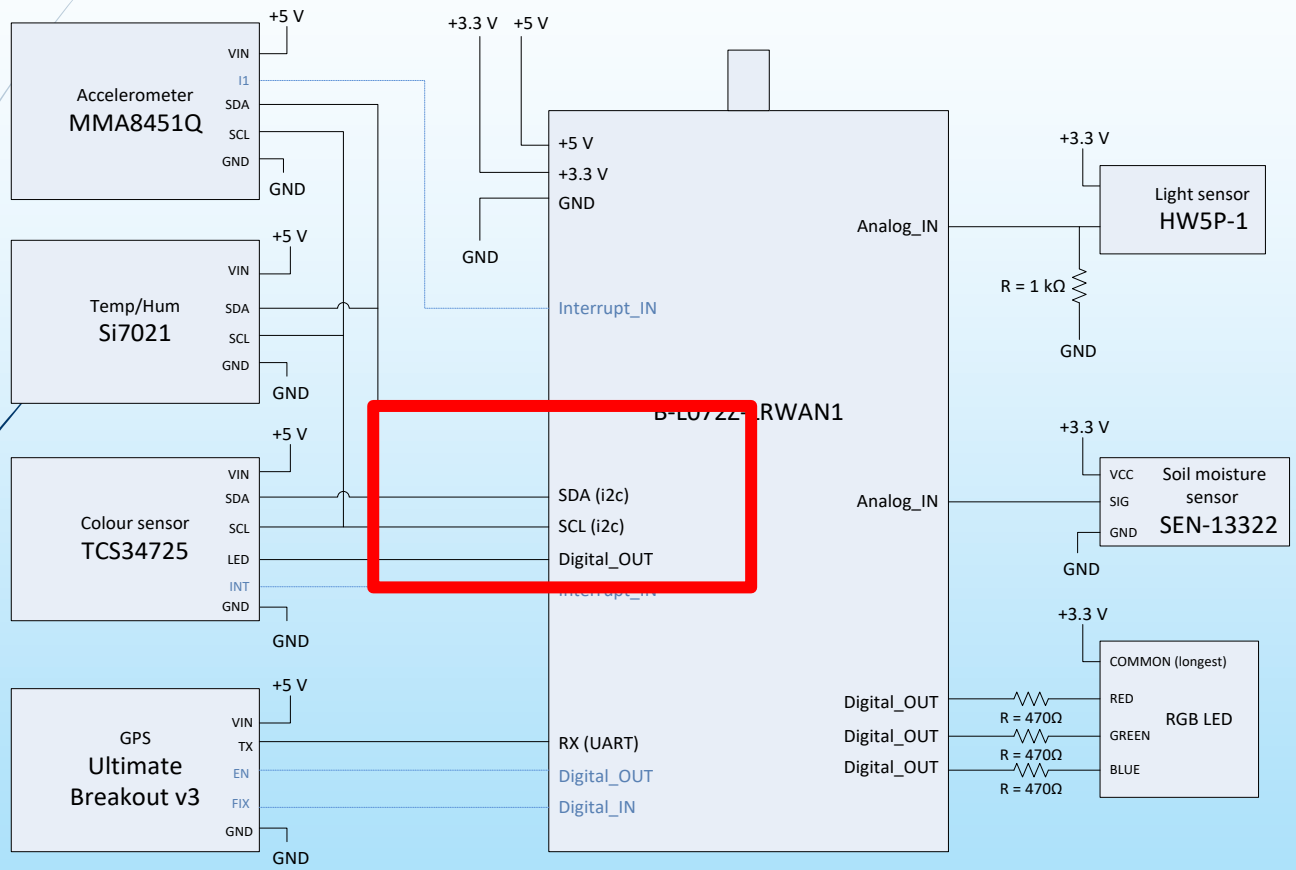
```
#include <ctype.h>  // toupper()
#include "mbed.h"

Serial  pc(USBTX, USBRX);

char c;

void pc_isr(void) {
  c = pc.getc();
}

int main() {
  pc.baud(115200);
  pc.format(8, pc.None, 1);
  pc.attach(pc_isr, pc.RxIrq);


  while(true) {
    if(c) {
      pc.printf("%c => %c\n", c,
                toupper(c));
      c = 0;
    }
  }
}
```

mbed I2C: I2C



- Master = B-L072Z-LRWAN1
- Slave(s) = I2C sensors

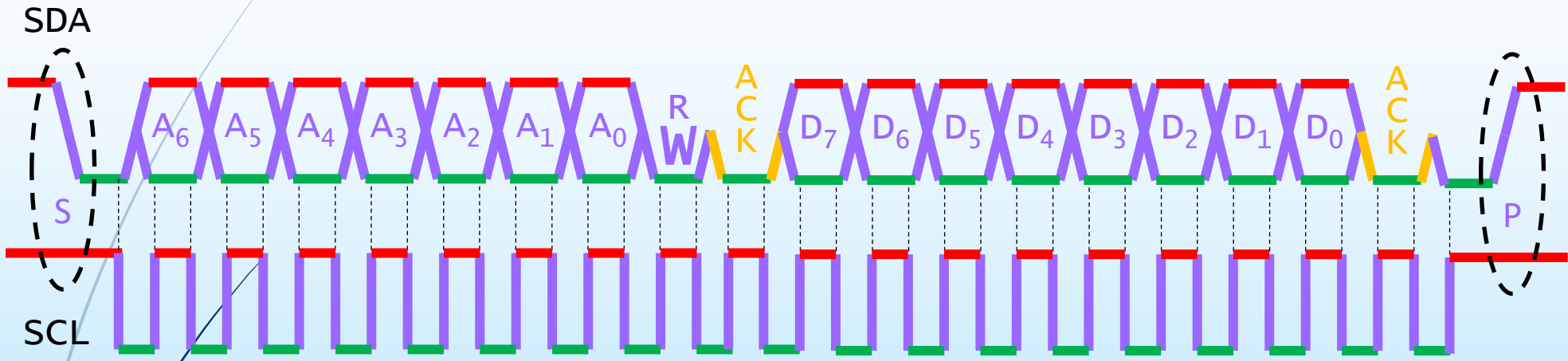
Block Diagram (Recall)



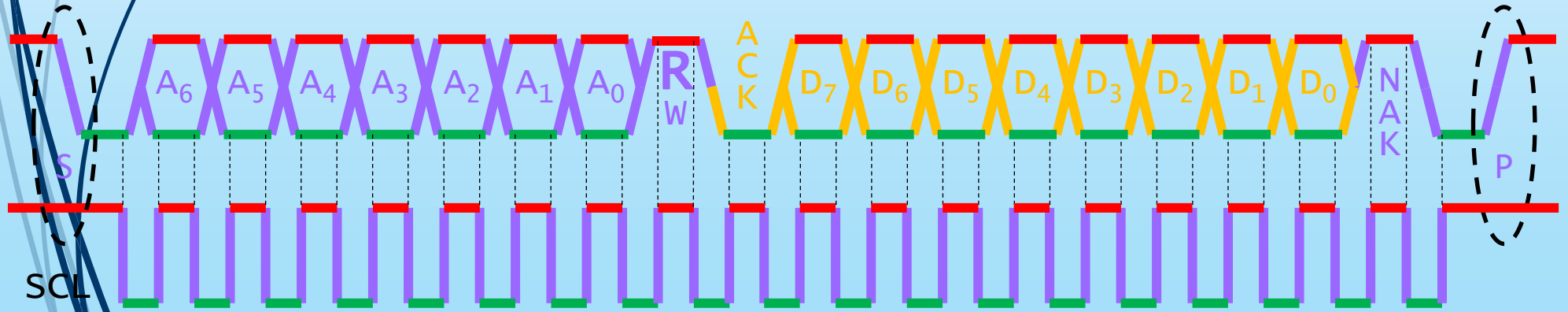
… worth considering that three of the sensors share the I2C bus

mbed I2C: I2C

- ‘write’ (transmitter) protocol



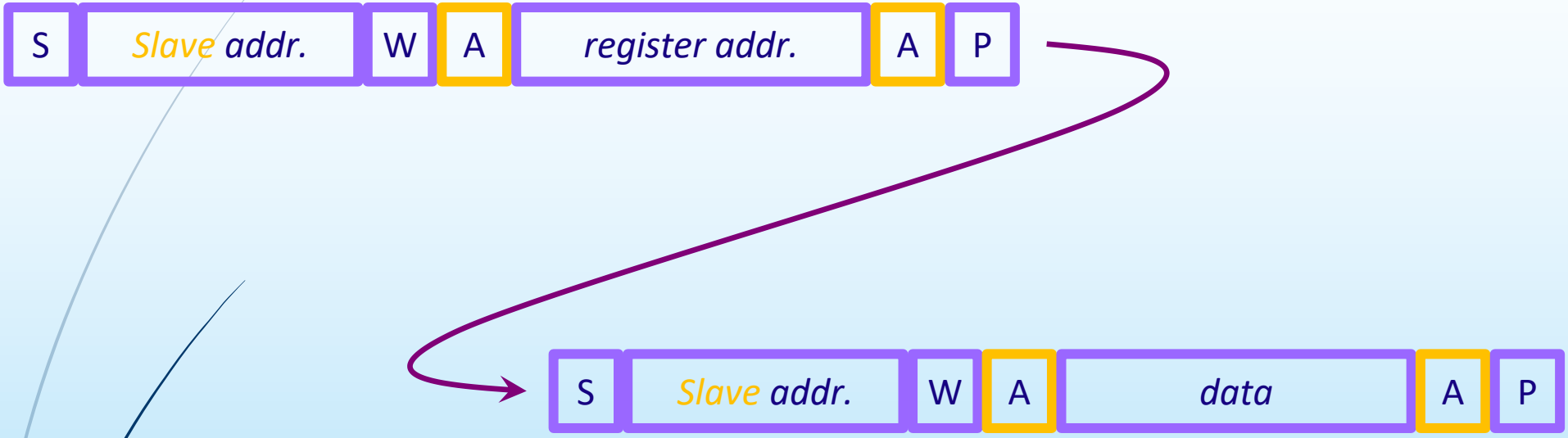
- ‘read’ (receiver) protocol



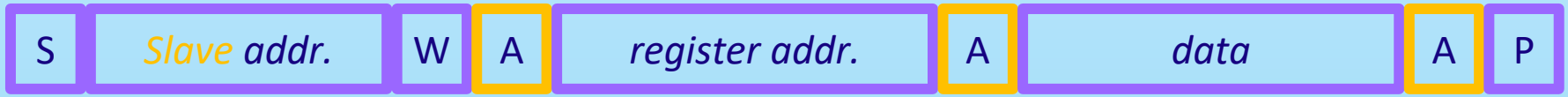
mbed I2C: I2C

- Slaves usually have registers to structure the info they store
- Each register is identied with an address within the slave
- Therefore, two addresses exist:
  - Slave address … for the I2C bus
  - Register address … for the slave
- To write into a register of a specific slave:
  - First, address the slave (write) and write the address of the register to be accessed
  - Then, address again the slave and write the data into the register previouly addressed.

mbed I2C: I2C

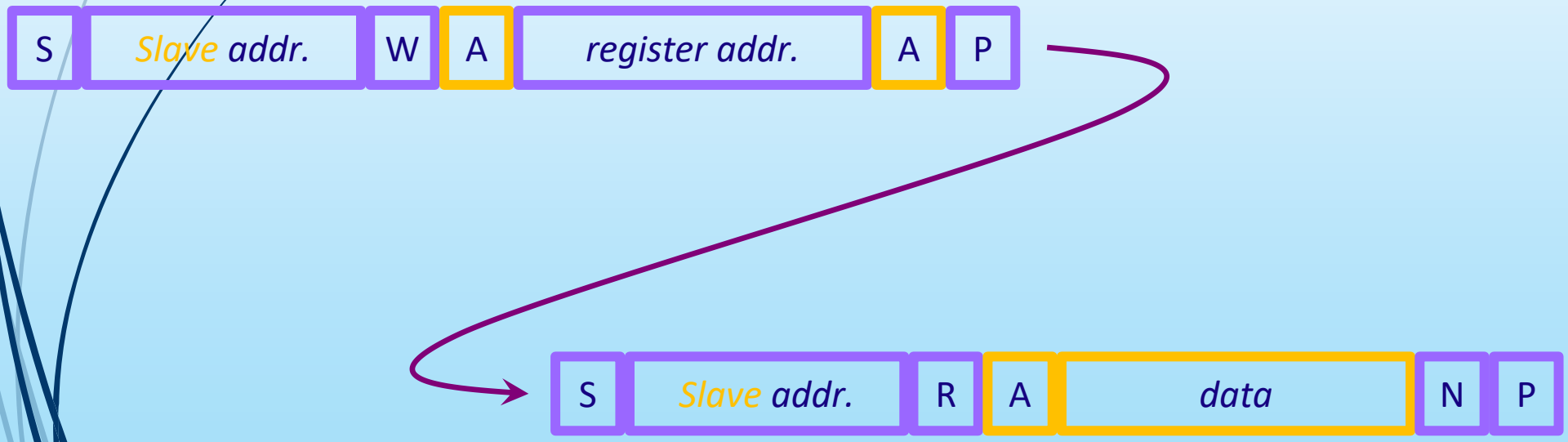


- Or some slaves join these two transactions into one:



mbed I2C: I2C

- To read from a register of a specific slave:
  - First, address the slave (write) and write the address of the register to be read
  - Then, address again the slave and read the data from the register previouly addressed.



mbed I2C: I2C

- *Master* I2C (simplified version)
- https://os.mbed.com/docs/mbed-os/v6.15/apis/i2c.html

```
I2C(PinName sda, PinName scl)
```

```
void frequency(int hz)      // defaults to 100 kHz
```

```
int read(int address, char *data, int length, bool repeated)
        // returns 0 if ACK, non-0 otherwise
```

```
int write(int address, const char *data, int length, bool repeated)
        // returns the number of written bytes, negative on error
```

```
// if repeated is true, the closing stop condition is omitted
// Defaults to false

// address must be the slave address << 1: mbed API implementation
issue
```

mbed I2C: Example

```
#include "mbed.h"
I2C      accel(p9, p10);
Serial   pc(USBTX, USBRX);
Ticker   ticker;
bool     tick_evnt;
void     ticker_isr(void) {
  tick_evnt = true;}

void main(void) {
  char          cmd[2];
  char          buf[1];
  float         ax;
  int           i;

  cmd[0] = 0x2a;
  cmd[1] = 0x01;
  accel.write(0x1d, cmd, 2);
  ticker.attach_us(ticker_isr,
                   1000000);
  pc.baud(115200);
```

```
  while(true) {
    if(tick_evnt) {
      cmd[0] = 0x01;
      accel.write(0x1d, cmd, 1);
      accel.read(0x1d, buf, 1);
      ax = buf[0]; //x-acceleration


      pc.printf("%4i s: %f g\n",
                i++, ax);
      tick_evnt = false;
    }
  }
}
```