mBED Basics

Jonathan Bachrach

EECS UC Berkeley

September 7, 2016
Last Time

- Talked about Basic C++
- C++ embedded library for ARM
- Object-oriented abstraction over microcontrollers
- Real Time OS
Basic Concepts

- Wrap objects around peripherals
- Initialize with pins using PinNames
- Get/Set attributes
- Read/Write values
- Extra control functions
- mbed namespace (used implicitly)

```c
#include "mbed.h"
DigitalOut LED(PB_1, High);
LED = 1;
```
Categories

- Task Management (and Real Time)
- Input and Outputs
- Interfaces (and USB)
- (Networking and Communication)
- (Security)

https://docs.mbed.com/docs/mbed-os-api-reference/en
Simple Task API

- Wait
- Timer
- Time
- Ticker

// ...
int main () {
    for (;;) {
        led = !led;
        led = !led;
        wait(0.5);
    }
}

Control reset, start and stop

Read read, read_us, read_ms,

32bit timer so only up to seconds cause counting microseconds

```c
Timer t;

int main() {
    t.start();
    printf("Hello World!\n");
    t.stop();
    printf("The time taken was \%f seconds\n", t.read());
}
```
For attaching callback function after time
- Attach with `attach`
- Warning about short callback functions

```c
Ticker flipper;
DigitalOut led1(LED1);
DigitalOut led2(LED2);

void flip() { led2 = !led2; }

int main() {
    led2 = 1;
    flipper.attach(&flip, 2.0); // setup flipper to call flip after 2 seconds

    // spin in a main loop. flipper will interrupt it to call flip
    while(1) {
        led1 = !led1;
        wait(0.2);
    }
}
```
Inputs and Outputs

- DigitalIn/Out
- AnalogIn
- BusIn/Out
- PortIn/Out
- PwmOut
- InterruptIn
- **PinMode** – PullUp, PullDown, PullNone
- **PinName** – look at arduino map below
- Constructor with PinName and PinMode
- PinMode with `mode`
- reader read plus overloaded dereferencer

```c
DigitalIn button(PB_1, PullUp);
uint8 x = button;
```
DigitalOut 11

- Constructor with PinName
- PinMode
- writer

```cpp
DigitalOut led(PB_1, PullUp);
led = 1;
```
Constructor with PinName

float

reader

```c
AnalogIn temp(PA_2);
if (temp > 0.5) { ... }
```
Constructor with PinName

float

reader/writer

AnalogOut speaker(PA_2);
speaker = 0.5;
- Able to group a bunch of digital pins together
- Read/write with uint
- BusInOut can switch with `input()` and `output()`

```cpp
BusOut leds(PB_0, PB_7, PB_6, PB_1);
for (int i = 0; i < 16; i++)
    leds = i;
```
- Group digital pins according to microcontroller ports with mask
- Read/write with uint
- BusInOut can switch with `input()` and `output()`

```c
PortOut leds(PortB, 0xf);
for (int i = 0; i < 16; i++)
    leds = i;
```
- Constructor sets pin
- Write duty cycle with float and read back set value
- Can set period in ms or us

```c
PwmOut led(PB_0);
led.period_ms(20);
led = 0.5;
```
InterruptIn

- Constructor sets pin to set interrupt
- Set call back function on particular event (rise, fall)
- Can read back value
- Enable/Disable Irq with disable_irq()

```cpp
InterruptIn button(PB_0);
DigitalOut led(PB_7);
DigitalOut led(PB_6);
void trigger () {
    rled = !rled;
}
int main () {
    button.rise(&trigger);
    while (true) {
        bled = !bled;
        wait(0.25);
    }
}
```
Serial
- ( I2C / I2CSlave )
- ( SPI / SPISlave )
- ( CAN )
- ( USB* )
Serial

- Constructor sets Tx and Rx pins
- Set baudrate with baud
- Configure with format for bits, parity, stop bits
- Default settings 9600 8N1
- getc for input and putc, printf for output
- Blocking writes bad for real-time

```c
Serial pc(USBTX, USBRX);
int main() {
    pc.printf("Hello World!\n");
    while(1) {
        pc.putc(pc.getc() + 1);
    }
}
```
Serial Polling

- Can poll char with readable
- Determine if write space with writeable

```c
Serial pc(USBTX, USBRX); // tx, rx
Serial device(p9, p10); // tx, rx

int main() {
  while(1) {
    if (pc.readable()) {
      device.putc(pc.getc());
    }
    if (device.readable()) {
      pc.putc(device.getc());
    }
  }
}
```
Serial Interrupts

- Attach callback with `attach` for rx or tx ready

```c
DigitalOut led1(LED1);
DigitalOut led2(LED2);

Serial pc(USBTX, USBRX);

void callback() {
    // Note: you need to actually read from the serial to clear the RX interrupt
    printf("%c\n", pc.getc());
    led2 = !led2;
}

int main() {
    pc.attach(&callback);
    while (1) {
        led1 = !led1;
        wait(0.5);
    }
}
```
This is what you need to get started
There is info on mbed website but
Most information is in mbed headers
Platform specific stuff is in target subdir

# basic headers
MBED/hal/api/
# target specific headers
MBED/hal/targets/hal/TARGET_STM/TARGET_STM32L4/TARGET_NUCLEO_L432KC
- Basic PCBs and Soldering
- Will cover more of mBED API later
• Section Monday 4-5p in Jacobs 210
• Get OtherMill Online training
• Start thinking partners
References

- STM32L432 Data Sheet
- Nucleo32 L432
- mBED APIs
  https://docs.mbed.com/docs/mbed-os-api-reference/en