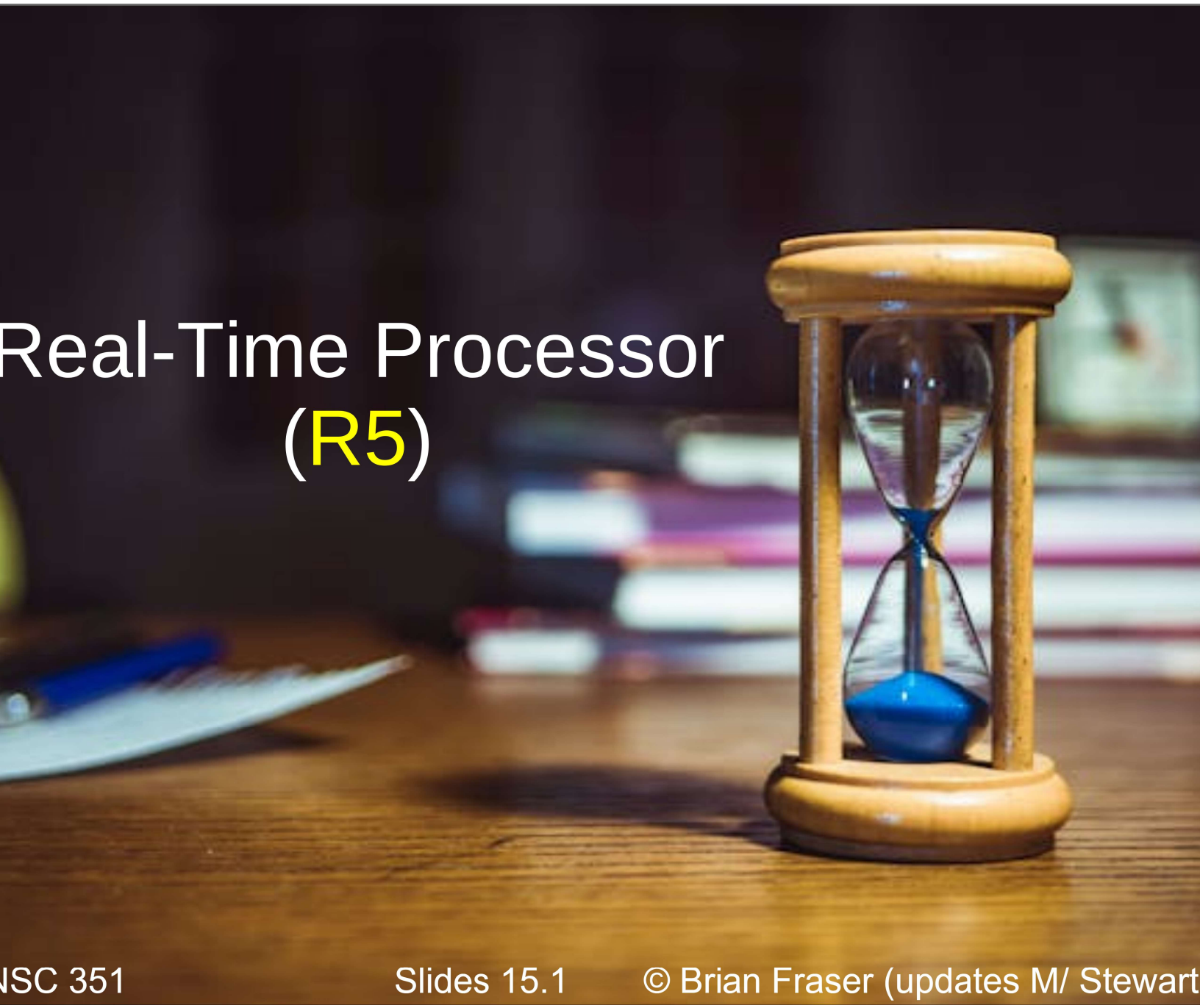


pics

- How can we do hard real-time on the BYAI?
- How can we get our code into the R5?
- How can we use GPIO with the R5?

Hard Real-Time with BYAI:
About the R5

r Definitions of Realtime

ard Real-time

- User requires..
- "Guaranteed Services"
  Mathematical/logical proof or exhaustive simulation required
- Hard real-time is about..

oft Real-time

- User only requires..

  (statistical analysis)
- "Best effort Services"
  E.g.: Average # missed deadline < 2 per minute.
- Soft real-time is about..

tivation

nux can do fine with soft real-time tasks
- ~10s of ms accuracy/latency

ard real-time
- Cannot use Linux:

  ..

  but, Linux gives us great software power!
- Analyze system and

  ..

  from soft real-time and non-real-time

ard real-time task solution
- Run hard real-time tasks on a processor without Linux
  - Use external processor (Arduino)
  - Use internal microcontroller (R5)

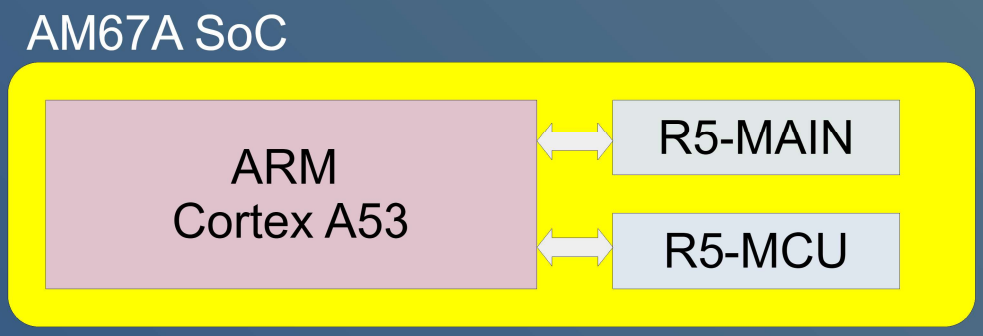
F Sub System

he BeagleY-AI has a
exas Instruments AM67A SoC..

- Powerful main processor:
  Arm Cortex-A53 (quad-core, 1.4GHz)
- 2 Dedicated real-time processors:
  Arm Cortex-R5 (single-core, 800MHz)
- 2 general purpose digital signal processors (DSPs):
  C7x DSP (4-TOPS with Matrix Multiply Accelerator)

e'll use:

- ..
- ..
  GPIO for NeoPixel LED



5 is useful for hard real-time interactions, such as:

- deterministic latency to respond to GPIO event
- .. GPIO and ..
  (Ex: timing ultrasonic distance sensor readings)
- .. (manual) protocol implementation:
  UART, I2C, SPI, Neo-pixel 1-wire protocol, etc.

alled the Cortex R5F Subsystem (R5FSS)

- The *F* means it has a hardware floating point unit.

Architecture

R5’s:

- Dedicated R5 core to manage boot, power mode
- Available R5 for running custom code.
- Available R5; able to access different GPIO pins.

ach R5:

- 32-bit ARM processors, 800MHz
- 2-banks of fast (uncached) memory:

  ..

esources

- Can share RAM banks with Linux
- Each R5F allows access to different GPIO pins
- Peripheral access (UART, etc)
- And more! (interrupts, ...)

erface to R5

nux's Remote Processor Framework

- ..
- called remoteproc

yai)$ **ls /sys/class/remoteproc/**

- remoteproc0 - DSP #1
- remoteproc1 - DSP #2
- remoteproc2 - R5 - MCU
- remoteproc3 - R5 - WKUP
- remoteproc4 - R5 - Main

iew status of processor by viewing its state file

(byai)$ cat /sys/class/remoteproc/rempoteproc3/state

Coding for the R5

mple Program

```
LAY_TIME_uS   (250 * 1000)

tree nodes for pin aliases
D0_NODE DT_ALIAS(led0)
N0_NODE DT_ALIAS(btn0)

st struct gpio_dt_spec led = GPIO_DT_SPEC_GET(LED0_NODE, gpios);
st struct gpio_dt_spec btn = GPIO_DT_SPEC_GET(BTN0_NODE, gpios);

d initialize_gpio(const struct gpio_dt_spec *pPin, int dir) {

ready_dt(pPin);
_configure_dt(pPin, dir);
```

```
int main(void) {
    printf("Hello World! %s\n");

    initialize_gpio(&led, GPIO_OUTPUT_A
    initialize_gpio(&btn, GPIO_INPUT);

    while (true) {
        gpio_pin_toggle_dt(&led);

        k_busy_wait(DELAY_TIME_uS);

       // Delay longer when button pres
        if (gpio_pin_get_dt(&btn) == 0)
            k_busy_wait(DELAY_TIME_uS);
        }
    }
    return 0;
}
```

ld Process

ur R5 programs will use..

etup

- Install the Zephyr SDK & Tools
  SDK = ..

n Host

- Build project using CMake
  Builds build/zephyr_mcu.elf
- Copy to NFS

n Target

- Copy zephyr_mcu.elf to /lib/firmware
- Load into R5:
  echo zephy_mcu.elf | sudo tee /sys/class/remoteproc/remoteproc2/firmwa
  echo start | sudo tee /sys/class/remoteproc/remoteproc2/state

GPIO with R5

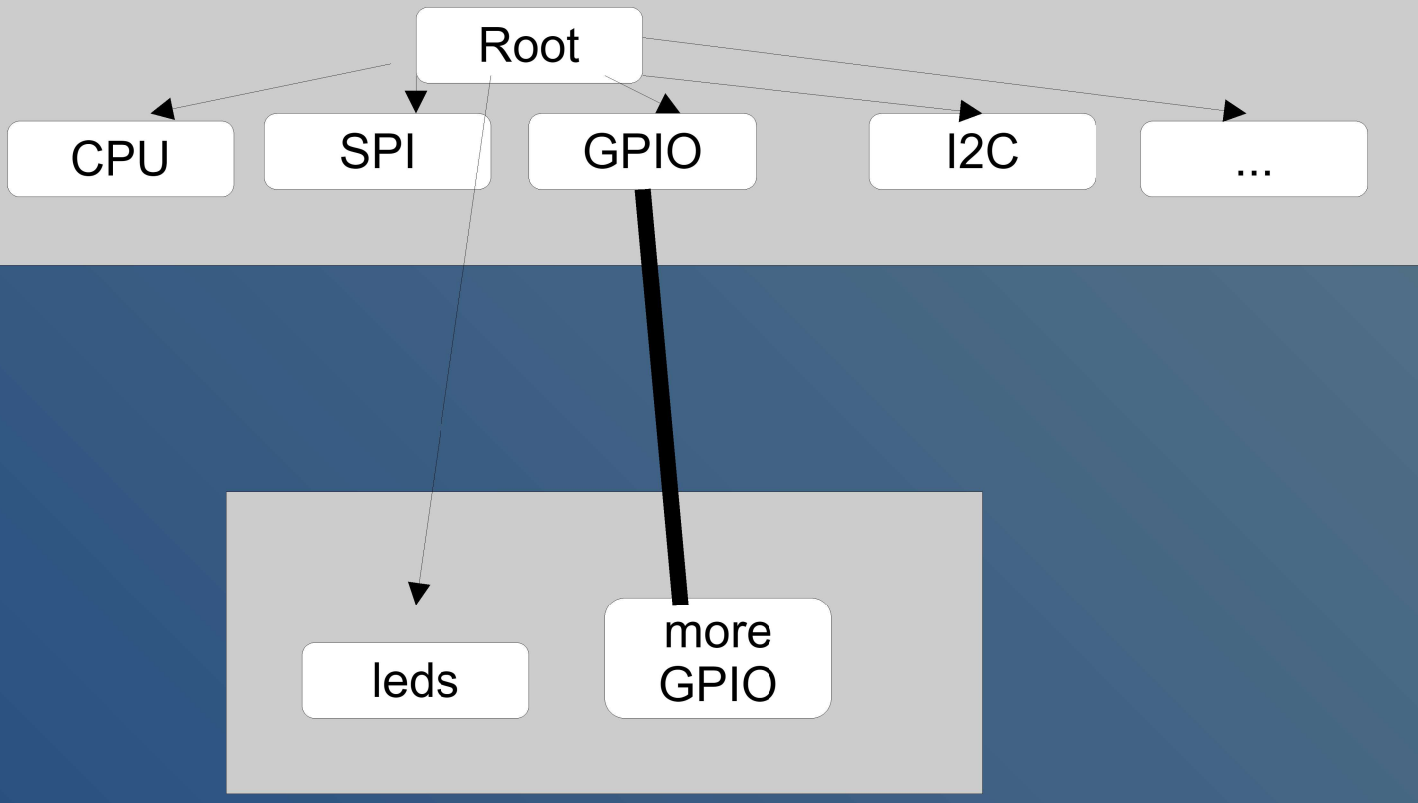
erview

code refers to pins by name.

```
vice tree nodes for pin aliases
ne LED0_NODE DT_ALIAS(led0)
c const struct gpio_dt_spec led = GPIO_DT_SPEC_GET(LED0_NODE, gpios);

ain(void)

pio_is_ready_dt(&led);
pio_pin_configure_dt(&led, GPIO_OUTPUT_ACTIVE);

hile (true) {
   gpio_pin_toggle_dt(&led);
   k_busy_wait(250 * 1000); //uS

eturn 0;
```

vice Tree

5f
Tree



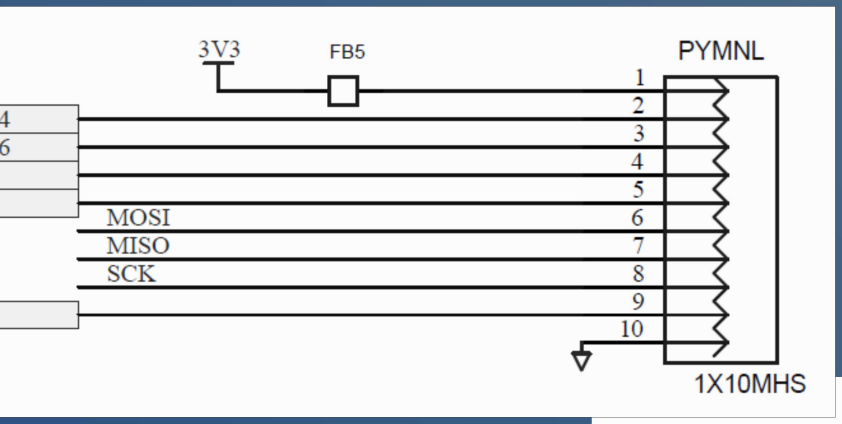
Tree

```
/ {
    aliases {
        led0 = &mcu_led;
    };

    leds {
        compatible = "gpio-leds";

        mcu_led: led_gpio7 {
            gpios = <&mcu_gpio0 9 GPIO_ACTIVE_HIGH>;
        };
    };

    pinctrl_mcu: pinctrl_mcu@4084000 {
        compatible = "ti,k3-pinctrl";
        reg = <0x04084000 0x88>;
        status = "okay";
    };
};

&pinctrl_mcu {
    mcu_gpio0_led_btn_default: mcu_gpio0_led_btn_default {
        pinmux = <K3_PINMUX(0x0024, PIN_OUTPUT, MUX_MODE_7)>; /* (B3)
    };
};

&mcu_gpio0 {
    pinctrl-0 = <&mcu_gpio0_led_btn_default>;
    pinctrl-names = "default";
    status = "okay";
};
```

mystifying GPIO

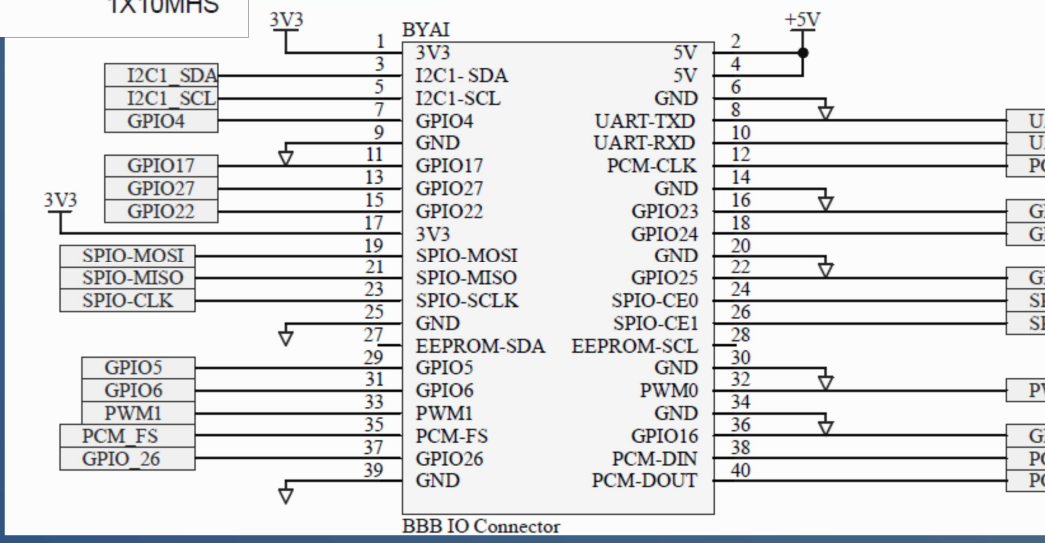
ow do we know what to put into a device tree?

- On schematic, identify pin:



SPI0-CE1 maps to hat-pin 26

is PYMNL-Pin 9
s to SPI0-CE1



n Hat Pinout

Pinout

Use		Use	Name	Connector		Name	Use		Hat Use	Expos
			3v3	1	2	5v				
elerometer, ADC, Audio		I2C1 - SDA	GPIO2	3	4	5v				LED S
elerometer, ADC, Audio		I2C1 - SCL	GPIO3	5	6	GND			UART Header	UART
	X		GPIO4	7	8	GPIO14	UART - TXD	X	UART - TXD	UART
			GND	9	10	GPIO15	UART - RXD	X	UART - RXD	UART
ary Encoder - EQEP0_B			GPIO17	11	12	GPIO18	PCM - CLK		Audio - BCLK	
) - DC			GPIO27	13	14	GND				LED S
) - RST			GPIO22	15	16	GPIO23		X	NeoPixel Data Out (R5 GPIO)	LED S
			3v3	17	18	GPIO24			Encoder Push-button (R5 GPIO)	
) - DIN		SPI0 - MOSI	GPIO10	19	20	GND				
	X?	SPI0 - MISO	GPIO9	21	22	GPIO25			LED	
) - CLK		SPI0 - SCLK	GPIO11	23	24	GPIO8	SPI0 - CE0		LCD - CS	
			GND	25	26	GPIO7	SPI0 - CE1	X		GPIO
y I2C, but prefer I2C1>		EEPROM - SDA	GPIO0	27	28	GPIO1	EEPROM-SCL		*<only I2C, but prefer I2C1>*	
stick Push-button			GPIO5	29	30	GND				
	X		GPIO6	31	32	GPIO12	PWM0		LED Emitter	
) - Backlight (BL)		PWM1	GPIO13	33	34	GND				
io - WCLK		PCM - FS	GPIO19	35	36	GPIO16			Rotary Encoder - EQEP0_A	
io - Reset			GPIO26	37	38	GPIO20	PCM - DIN		Audio - DIN	
			GND	39	40	GPIO21	PCM - DOUT		Audio - DOUT	
	X	Available GPIO								

eckout info on Hat Pin

se website to investigate a hat pin
tps://pinout.beagleboard.io/

MCU | GPIO | SoC Pin | 1-WIRE | I2C | 5v Power | Ground | GPCLK* | JTAG*
PWM | 3v3 Power | UART | SPI | DPI* | PCM | SDIO*

Browse pinouts for HATs, pHATs and add-ons »

GPIO 7 (SPI Chip Select 1)

Alt0	Alt2	Alt7
WKUP_UART0_RXD	MCU_SPI0_CS2	MCU_GPIO0_9

- Physical/Board pin 26
- GPIO/BCM pin 7
- SoC pin B3

D Config Register

/e know we are working with Ball Number B3

- Look it up in processor data sheet: https://www.ti.com/lit/ds/symlink/am67a.pdf

BALL UMBER [1]	BALL NAME [2] PADCONFIG Register [15] PADCONFIG Address [16]	SIGNAL NAME [3]
B3	WKUP_UART0_RXD PADCONFIG MCU_PADCONFIG9 0x04084024	WKUP_UART0_RXD
		MCU_SPI0_CS2
		MCU_GPIO0_9

ells us:

- .. (**MCU**_PADCONFIG9)
- Configuration reg addr = 0x04084024

nfig Register Offset

iven the config register address,
e actually need..

- The pin's config register offset is bottom 13 bits of the addre

```
_PINMUX(offset, value, mux_mode)  (((offset) & 0x1fff)) ((value) | (mux_
```

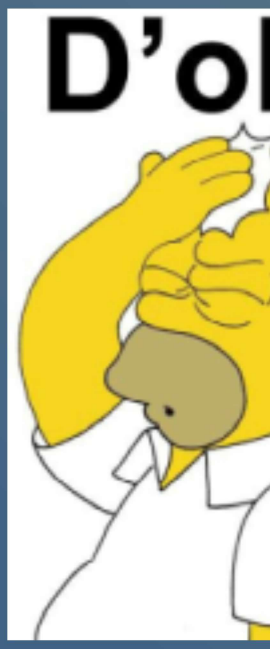
- So offset for address 0x04084024
  ..

```
rl_mcu {
u_gpio0_led_btn_default: mcu_gpio0_led_btn_default {
  pinmux = <K3_PINMUX(0x0024, PIN_OUTPUT, MUX_MODE_7)>; /* (B3) GPIO
```

nfiguration Confession

his investigation is (likely?)correct;
however, the configuration is not
yet working for MCU GPIO!

- A work-around is to
execute the GPIO command in Linux to set configuration:
gpioset gpiochip0 9=1
- This causes Linux to setup the configuration
in the way we need it for the R5.



Example
Blinky

e up

et’s have R5 flash a custom wired LED

- Wire this circuit

ocess: Build

uild on host

```
cmake -S . -B build -DBOARD=beagley_ai/j722s/mcu_r5f0_0
cd build
make
cd ..
```

opy to NFS

```
mkdir -p ~/cmpt433/public/r5/
cp build/zephyr/zephyr.elf ~/cmpt433/public/r5/zephyr_mcu.elf
```

ocess: Install

opy firmware from NFS to firmware folder

```
sudo cp /mnt/remote/r5/zephyr_mcu.elf /lib/firmware/
```

tart code

```
echo zephyr_mcu.elf | sudo tee /sys/class/remoteproc/remoteproc2/firmware
echo start | sudo tee /sys/class/remoteproc/remoteproc2/state
```

/ork-around for GPIO Config

```
gpioset gpiochip0 9=1
```

mmary

se R5 to meet hard real-time deadlines for RT tasks

rocess

- Develop on host
- Compile on host, install on Target

PIO for R5

- Use of device trees
- Finding pin info:
  - Use circuit diagram to find pin.
  - Use BeagleY-AI pinout for ball number.
  - Use AM67a Datasheet for register info.
  - Configure in device tree.