dht11 device driver

工海系 b06501018 朱紹勳

DHT11

- ftrace
- Trace code & Explaination
  - plateform device & platform driver
    - define & register, management
  - probe
    1. Initialize & allocate
    2. Attribute
       info
       mode
       type
       channel
    3. read raw data
- Summary of driver program

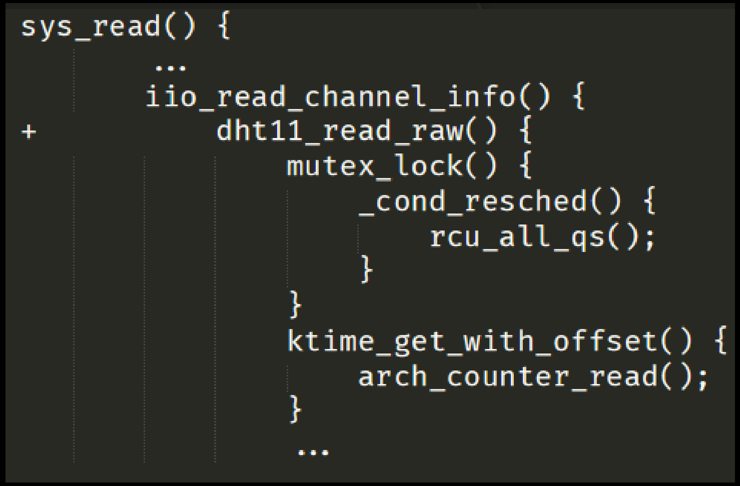
ftrace

- A tracing framework for Linux kernel
  - static trace point
  - dynamic trace point
- For simplicity we can install a CMD tool of ftrace – “trace-cmd”

```
sudo apt install trace-cmd
sudo trace-cmd record –p function_graph –F bash dht11_info.sh
```

(press Ctrl+C to terminate and generate report)

```
trace-cmd report
```

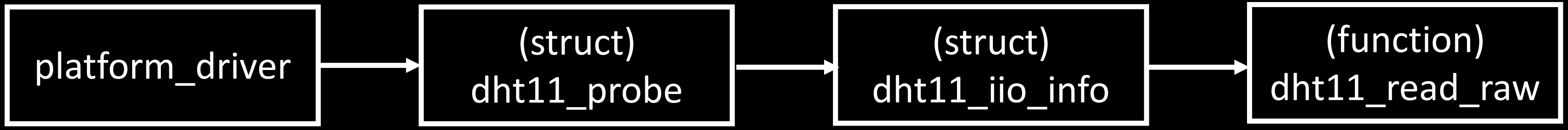


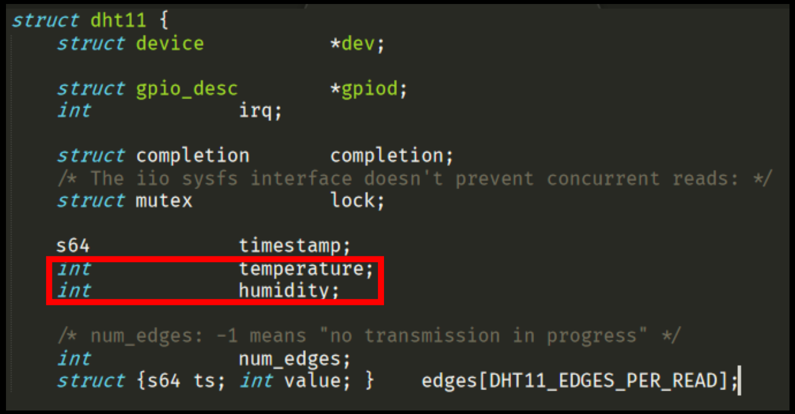
```
sys_read() {
        ...
        iio_read_channel_info() {
+           dht11_read_raw() {
                mutex_lock() {
                    _cond_resched() {
                        rcu_all_qs();
                    }
                }
                ktime_get_with_offset() {
                    arch_counter_read();
                }
                ...
```

Summary of iio driver program

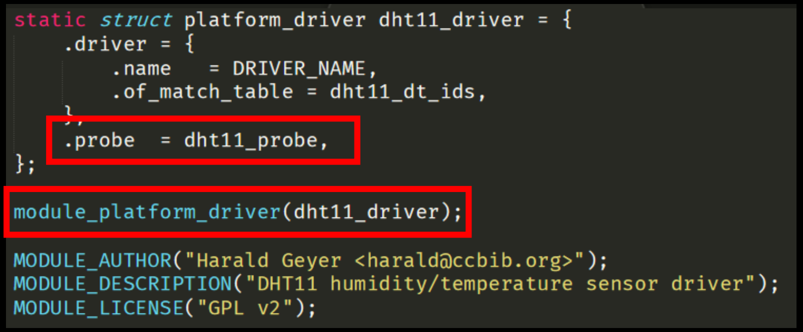
- 1. Define iio_dev
- 2. Set up channel
- 3. implement read / write
- 4. iio_info
- 5. implement and register iio_dev

• The calling sequence of accessing dht11 data

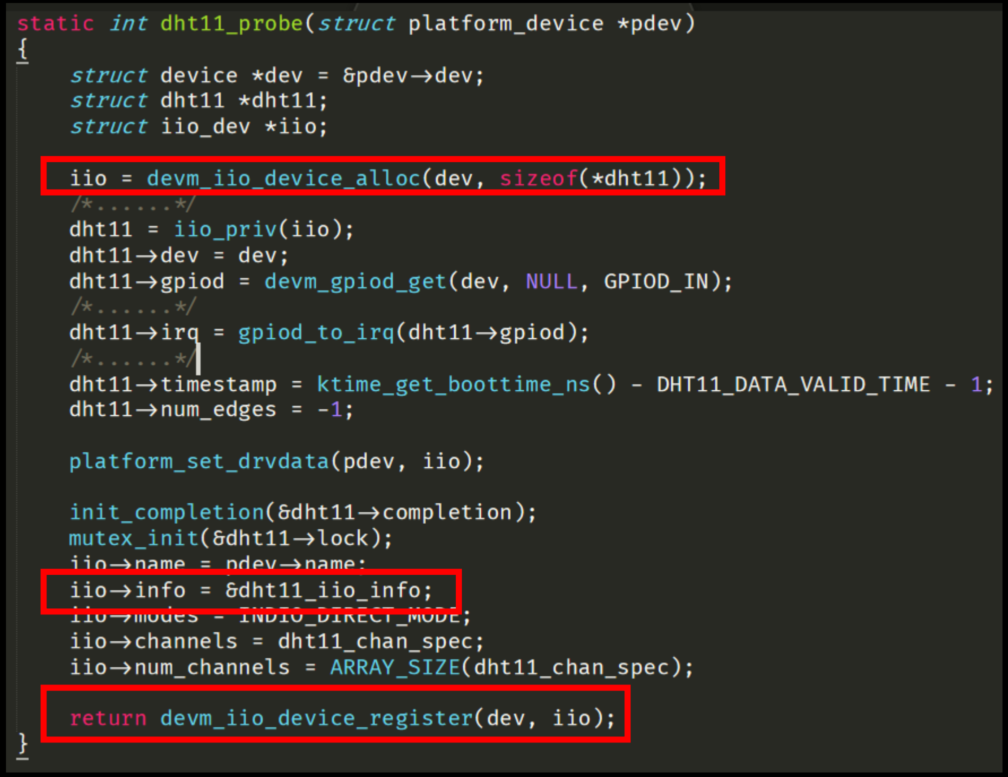




```
struct dht11 {
	struct device			*dev;

	struct gpio_desc		*gpiod;
	int				irq;

	struct completion		completion;
	/* The iio sysfs interface doesn't prevent concurrent reads: */
	struct mutex			lock;

	s64				timestamp;
	int				temperature;
	int				humidity;

	/* num_edges: -1 means "no transmission in progress" */
	int				num_edges;
	struct {s64 ts; int value; }	edges[DHT11_EDGES_PER_READ];
```

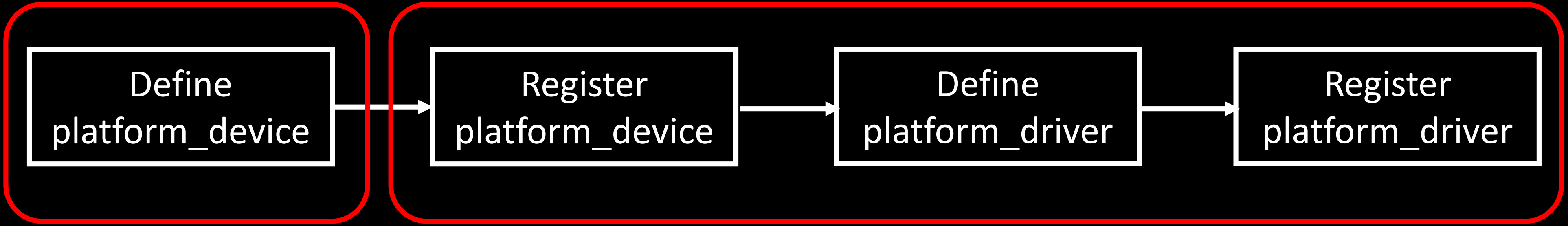


```
static struct platform_driver dht11_driver = {
	.driver = {
		.name	= DRIVER_NAME,
		.of_match_table = dht11_dt_ids,
	},
	.probe	= dht11_probe,
};

module_platform_driver(dht11_driver);

MODULE_AUTHOR("Harald Geyer <harald@ccbib.org>");
MODULE_DESCRIPTION("DHT11 humidity/temperature sensor driver");
MODULE_LICENSE("GPL v2");
```



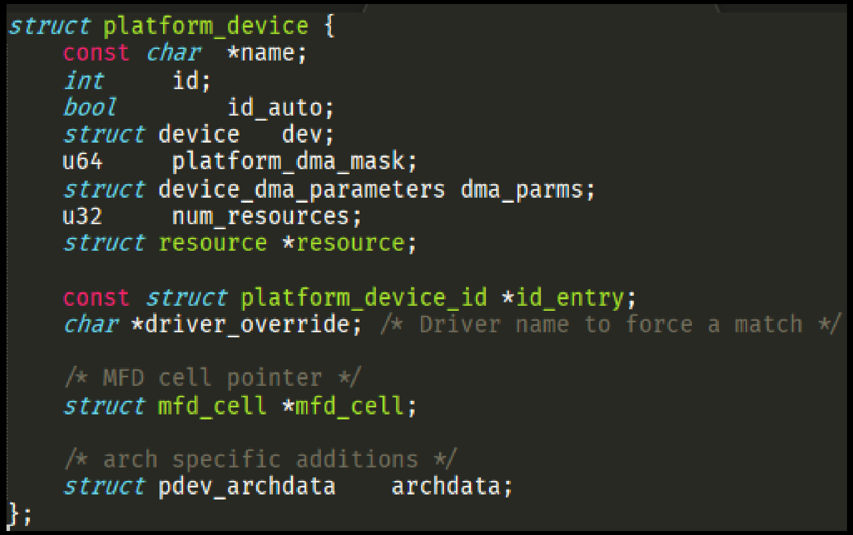
```
static int dht11_probe(struct platform_device *pdev)
{
	struct device *dev = &pdev->dev;
	struct dht11 *dht11;
	struct iio_dev *iio;

	iio = devm_iio_device_alloc(dev, sizeof(*dht11));
	/*......*/
	dht11 = iio_priv(iio);
	dht11->dev = dev;
	dht11->gpiod = devm_gpiod_get(dev, NULL, GPIOD_IN);
	/*......*/
	dht11->irq = gpiod_to_irq(dht11->gpiod);
	/*......*/
	dht11->timestamp = ktime_get_boottime_ns() - DHT11_DATA_VALID_TIME - 1;
	dht11->num_edges = -1;

	platform_set_drvdata(pdev, iio);

	init_completion(&dht11->completion);
	mutex_init(&dht11->lock);
	iio->name = pdev->name;
	iio->info = &dht11_iio_info;
	iio->modes = INDIO_DIRECT_MODE;
	iio->channels = dht11_chan_spec;
	iio->num_channels = ARRAY_SIZE(dht11_chan_spec);

	return devm_iio_device_register(dev, iio);
}
```

Driver Management & Registration

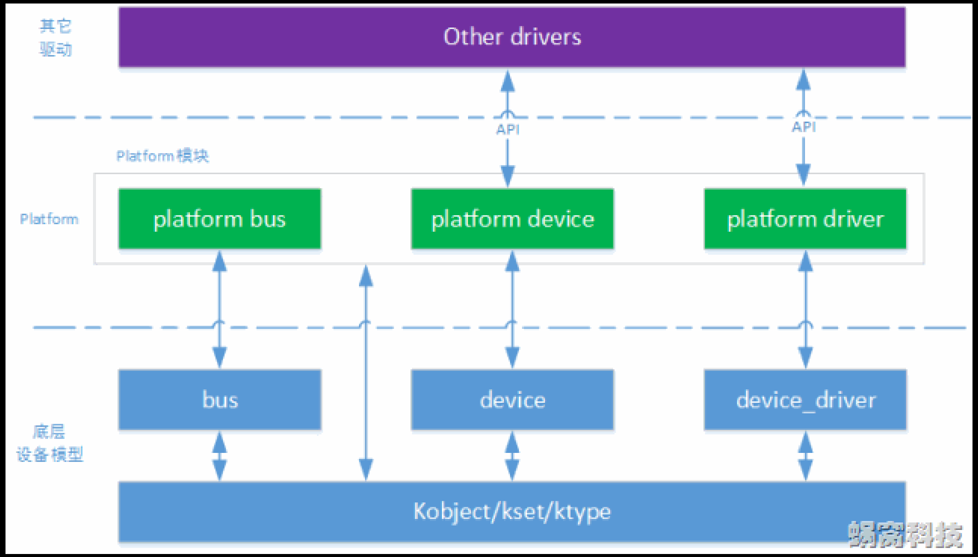


platform_device.h

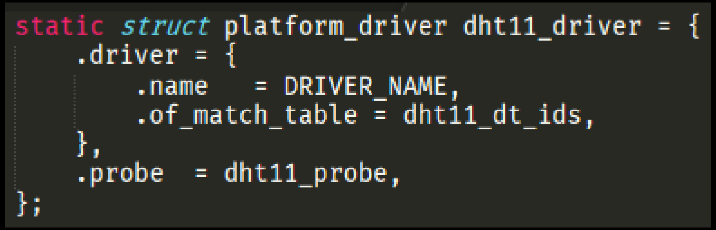
dht11.c



```
struct platform_device {
    const char  *name;
    int         id;
    bool            id_auto;
    struct device   dev;
    u64         platform_dma_mask;
    struct device_dma_parameters dma_parms;
    u32         num_resources;
    struct resource *resource;

    const struct platform_device_id *id_entry;
    char *driver_override; /* Driver name to force a match */

    /* MFD cell pointer */
    struct mfd_cell *mfd_cell;

    /* arch specific additions */
    struct pdev_archdata    archdata;
};
```



- Register platform_device
  - Under `dht_probe()` in `/dirver/iio/humidity/dht11.c`
    - `return devm_iio_device_register(dev, iio);`
      - dev : platform device
      - iio : Generate by `devm_iio_device_alloc(dev, sizeof(*dht11));`
        Initialize with name, info, modes, channels…
    - The "devm" is related to device resource management, in <devres.c>
      - Register & call
      - ex: gpio, irq
- Define platform_driver

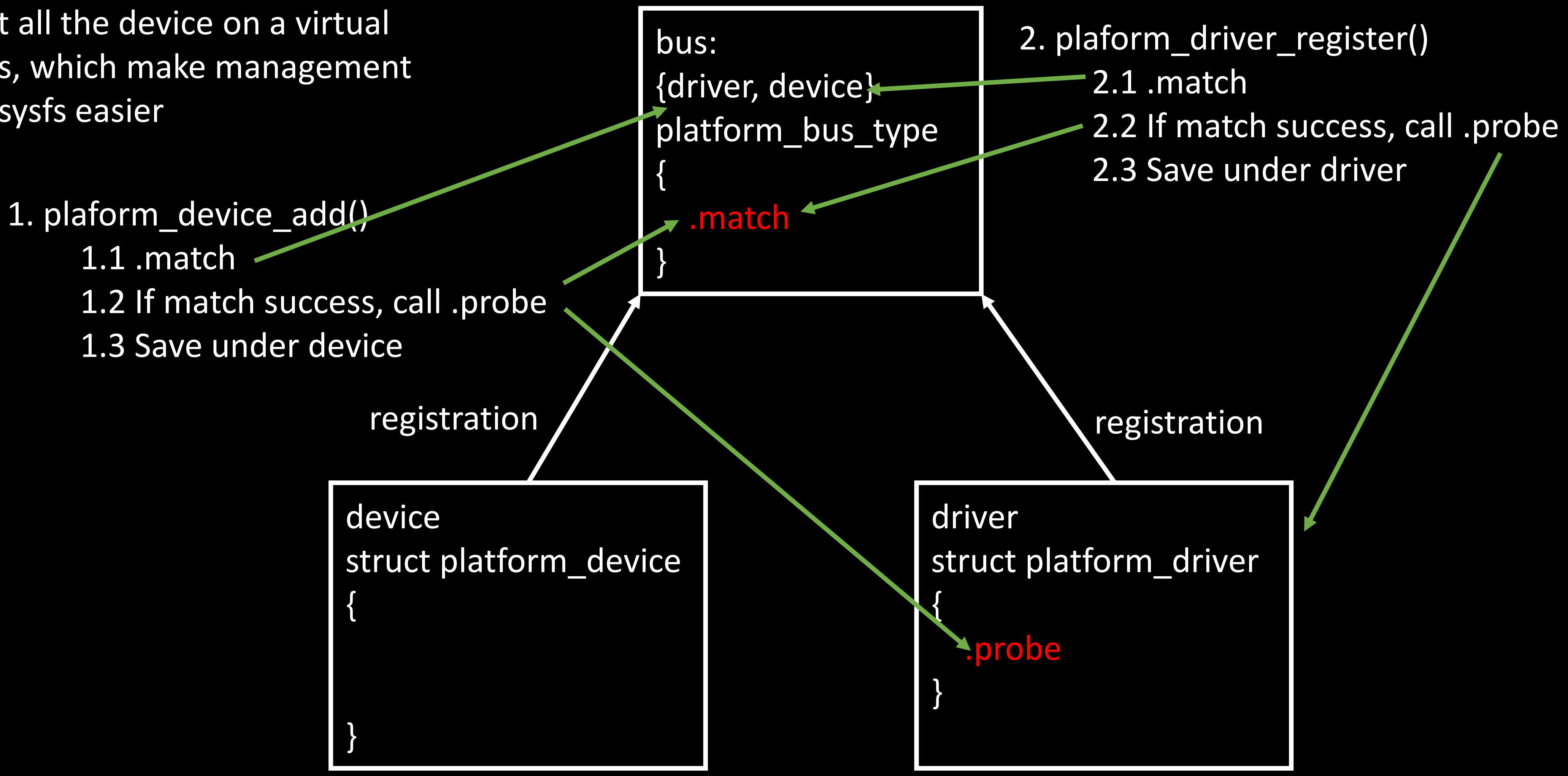


```c
static struct platform_driver dht11_driver = {
	.driver = {
		.name	= DRIVER_NAME,
		.of_match_table = dht11_dt_ids,
	},
	.probe  = dht11_probe,
};
```

- Register platform_driver
  - `module_platform_driver(dht11_driver);`
    - The module_platform_driver() will pass platform_structure to another macro `platform_driver_register()`, then to `__platform_driver_register()`, which responsible for register routine

Platform & Seperation

Put all the device on a virtual bus, which make management of sysfs easier



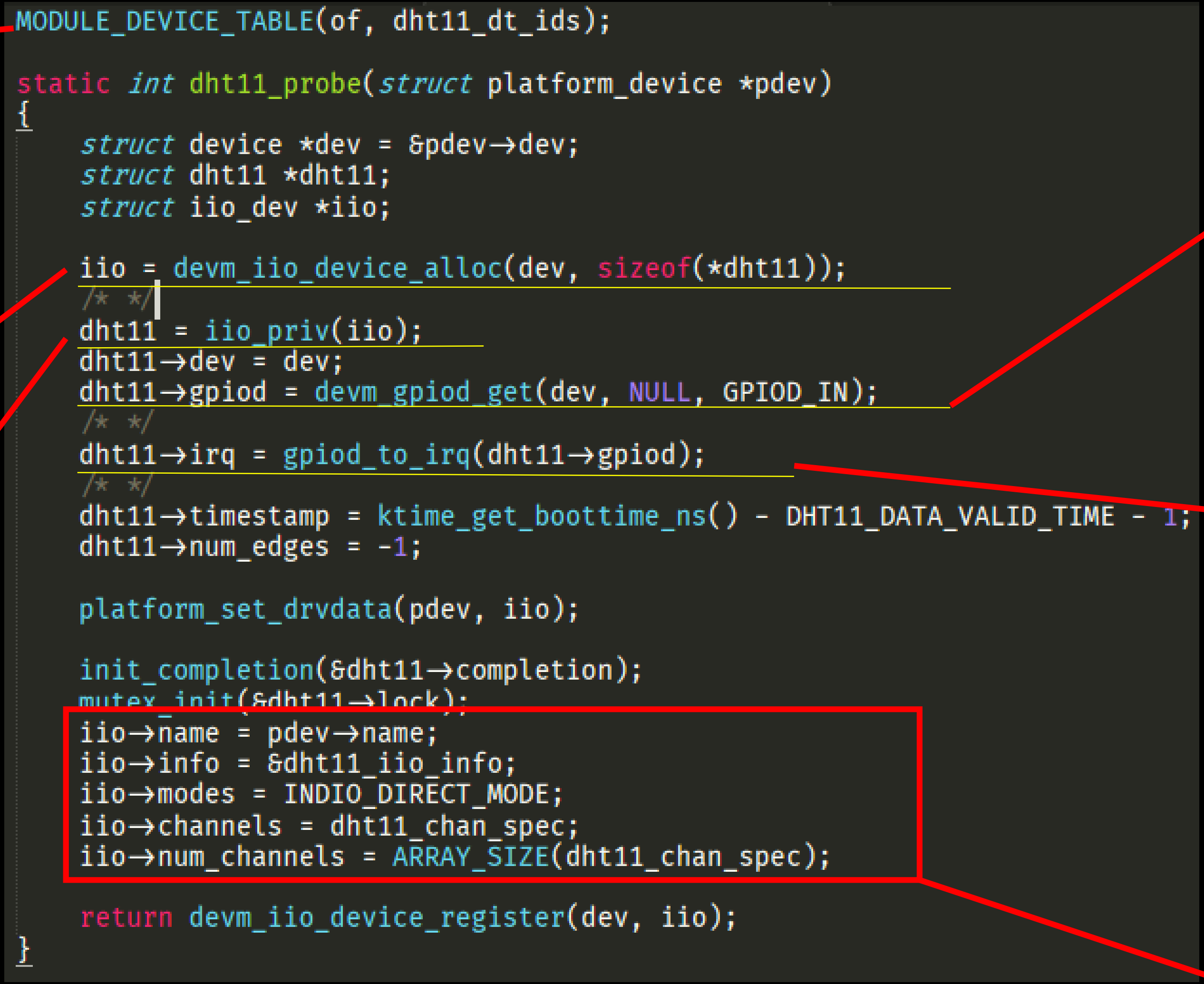
Probe

- Test the functionality and finish the last step of registration
- Probe() is called when having same name of "device name" & "driver name" on bus
- Difference between `probe()` and `init()`
  - probe is used platform device driver, PCI, USB…
    the platform need to "match" before being called
  - probe also support hot-plugging

for hot plugging. It exports the table of supported devices, so when the kernel discovers a new device, the right module will be loaded for it.

allocates memory for an IIO device.

allocate space for dht11 variable, the private data of iio_dev will be filled in this structure



```
MODULE_DEVICE_TABLE(of, dht11_dt_ids);

static int dht11_probe(struct platform_device *pdev)
{
    struct device *dev = &pdev->dev;
    struct dht11 *dht11;
    struct iio_dev *iio;

    iio = devm_iio_device_alloc(dev, sizeof(*dht11));
    /* */
    dht11 = iio_priv(iio);
    dht11->dev = dev;
    dht11->gpiod = devm_gpiod_get(dev, NULL, GPIOD_IN);
    /* */
    dht11->irq = gpiod_to_irq(dht11->gpiod);
    /* */
    dht11->timestamp = ktime_get_boottime_ns() - DHT11_DATA_VALID_TIME - 1;
    dht11->num_edges = -1;

    platform_set_drvdata(pdev, iio);

    init_completion(&dht11->completion);
    mutex_init(&dht11->lock);
    iio->name = pdev->name;
    iio->info = &dht11_iio_info;
    iio->modes = INDIO_DIRECT_MODE;
    iio->channels = dht11_chan_spec;
    iio->num_channels = ARRAY_SIZE(dht11_chan_spec);

    return devm_iio_device_register(dev, iio);
}
```

GPIOs mappings are defined in the consumer device's node

return gpio descriptor

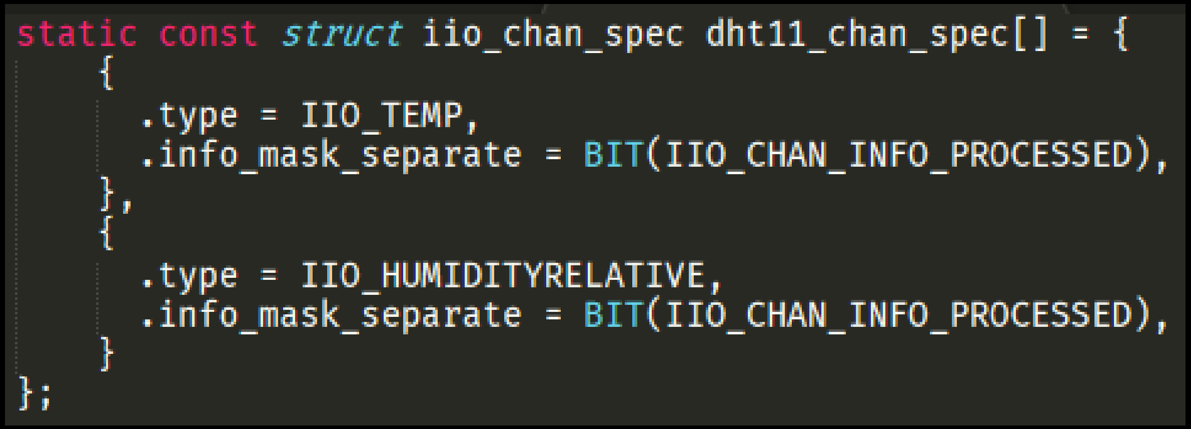
struct device *dev,
const char *con_id,
enum gpiod_flags flags

The gpiod_to_irq() will return the corresponding irq number

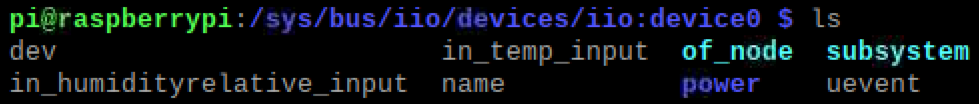
Initialize the attribute of iio_dev

channel, type, info_mask

- One channel represent a way of giving data. In dht11, there are temperature and humidity, so there will be two channels on dht11.

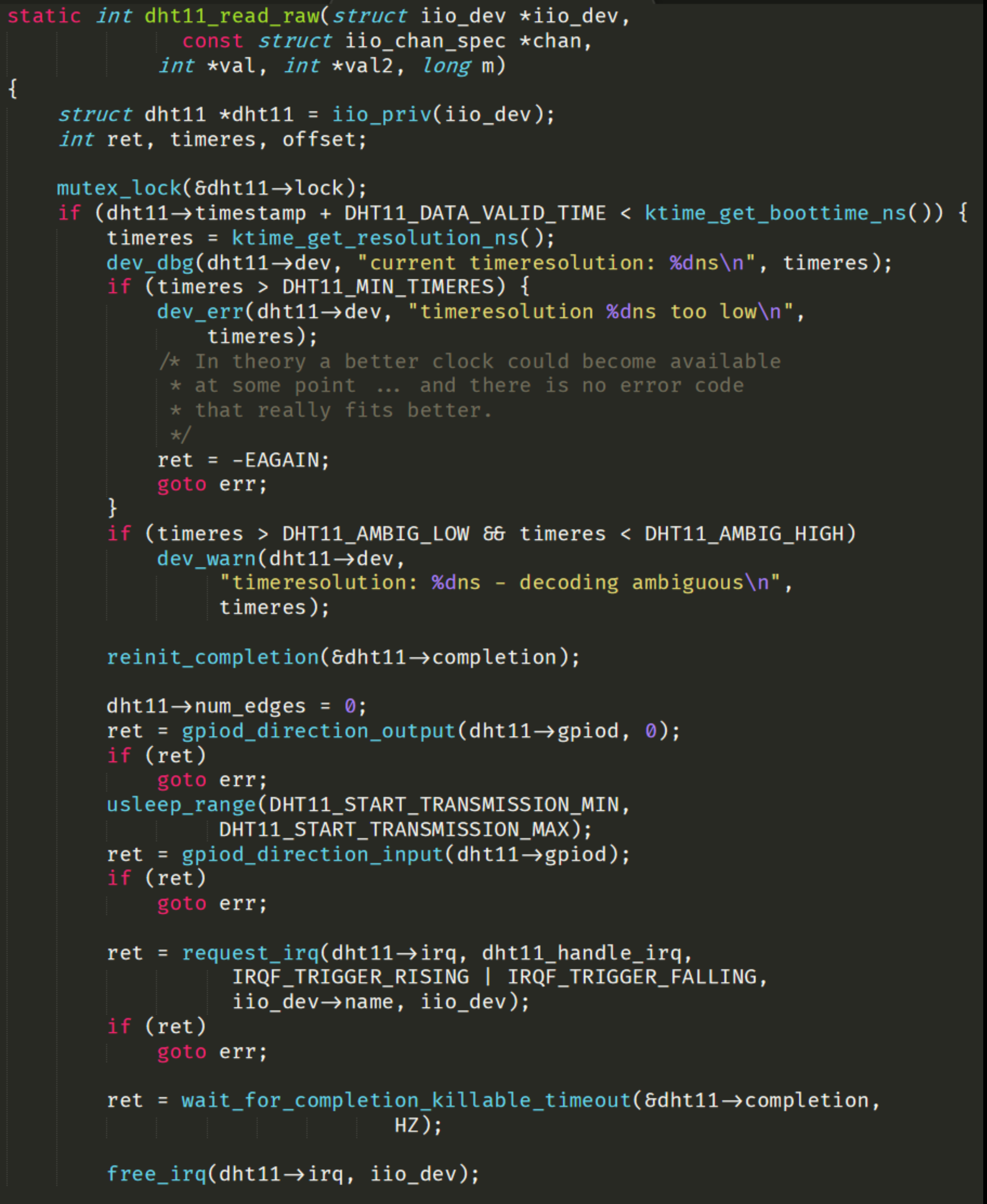


```
static const struct iio_chan_spec dht11_chan_spec[] = {
	{
		.type = IIO_TEMP,
		.info_mask_separate = BIT(IIO_CHAN_INFO_PROCESSED),
	},
	{
		.type = IIO_HUMIDITYRELATIVE,
		.info_mask_separate = BIT(IIO_CHAN_INFO_PROCESSED),
	}
};
```

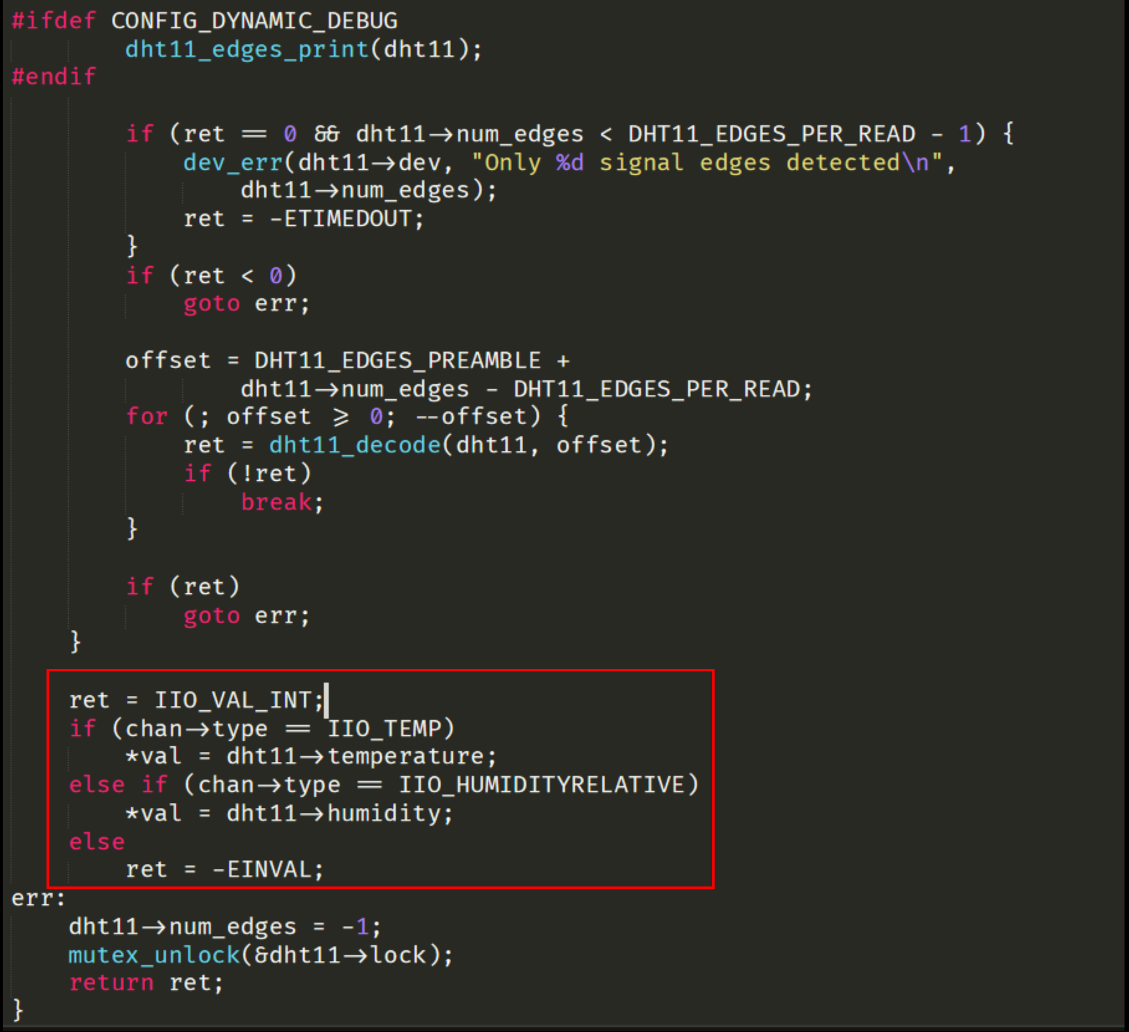


```
pi@raspberrypi:/sys/bus/iio/devices/iio:device0 $ ls
dev                           in_temp_input  of_node  subsystem
in_humidityrelative_input     name           power    uevent
```

- This will generate two node with the form of `in_XXX_input` under the path `sys/bus/iio/devices/iio:device`
- More over, if there are multiple value can be used to specific type, we can label them with `.indexed = n;`
- The `info_mask_sperpate` means attributes will be specific to channel



```c
static int dht11_read_raw(struct iio_dev *iio_dev,
              const struct iio_chan_spec *chan,
            int *val, int *val2, long m)
{
    struct dht11 *dht11 = iio_priv(iio_dev);
    int ret, timeres, offset;

    mutex_lock(&dht11->lock);
    if (dht11->timestamp + DHT11_DATA_VALID_TIME < ktime_get_boottime_ns()) {
        timeres = ktime_get_resolution_ns();
        dev_dbg(dht11->dev, "current timeresolution: %dns\n", timeres);
        if (timeres > DHT11_MIN_TIMERES) {
            dev_err(dht11->dev, "timeresolution %dns too low\n",
                timeres);
            /* In theory a better clock could become available
             * at some point ... and there is no error code
             * that really fits better.
             */
            ret = -EAGAIN;
            goto err;
        }
        if (timeres > DHT11_AMBIG_LOW && timeres < DHT11_AMBIG_HIGH)
            dev_warn(dht11->dev,
                 "timeresolution: %dns - decoding ambiguous\n",
                 timeres);

        reinit_completion(&dht11->completion);

        dht11->num_edges = 0;
        ret = gpiod_direction_output(dht11->gpiod, 0);
        if (ret)
            goto err;
        usleep_range(DHT11_START_TRANSMISSION_MIN,
                 DHT11_START_TRANSMISSION_MAX);
        ret = gpiod_direction_input(dht11->gpiod);
        if (ret)
            goto err;

        ret = request_irq(dht11->irq, dht11_handle_irq,
                  IRQF_TRIGGER_RISING | IRQF_TRIGGER_FALLING,
                  iio_dev->name, iio_dev);
        if (ret)
            goto err;

        ret = wait_for_completion_killable_timeout(&dht11->completion,
                                HZ);

        free_irq(dht11->irq, iio_dev);
```



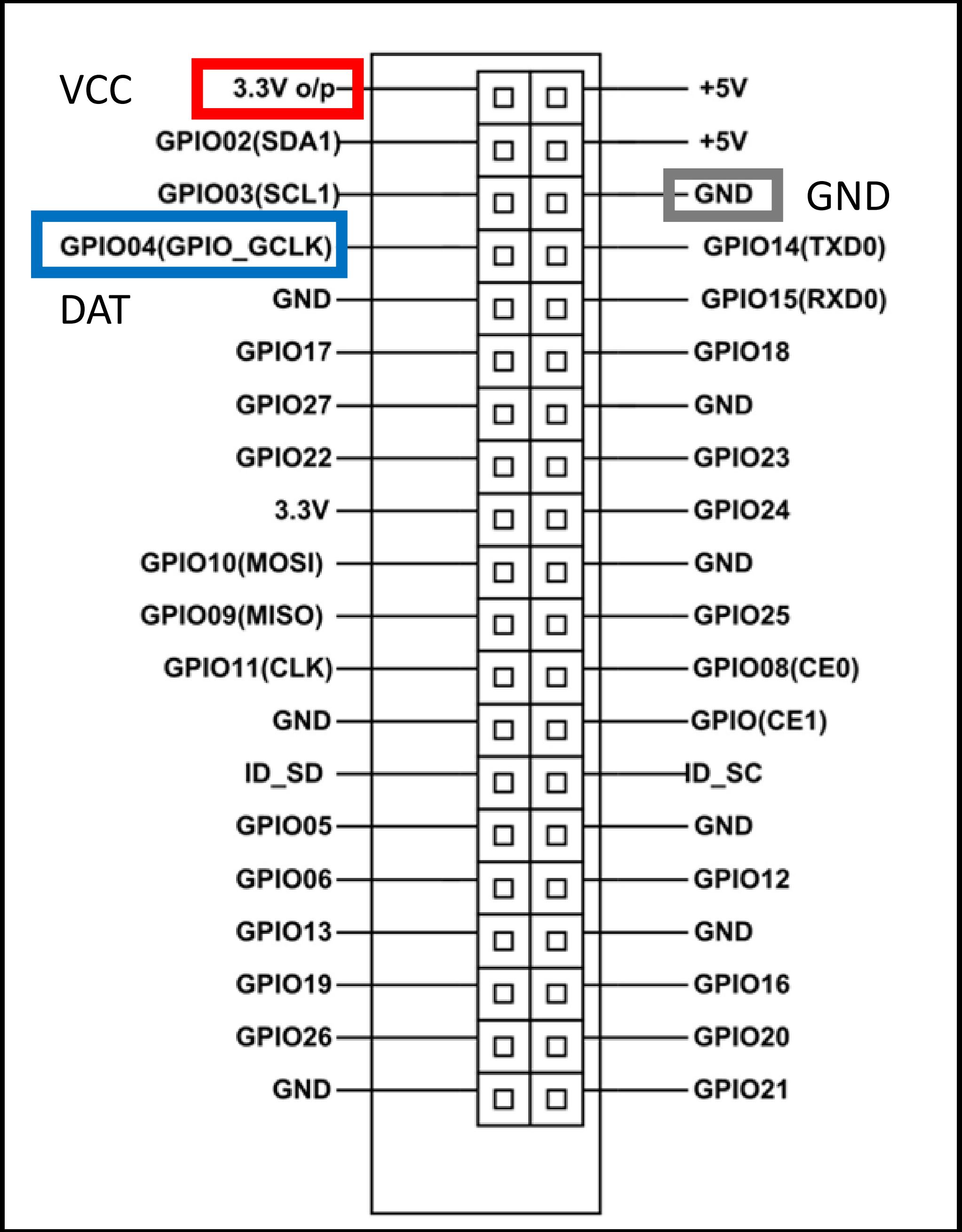
```c
#ifdef CONFIG_DYNAMIC_DEBUG
        dht11_edges_print(dht11);
#endif

        if (ret == 0 && dht11->num_edges < DHT11_EDGES_PER_READ - 1) {
            dev_err(dht11->dev, "Only %d signal edges detected\n",
                dht11->num_edges);
            ret = -ETIMEDOUT;
        }
        if (ret < 0)
            goto err;

        offset = DHT11_EDGES_PREAMBLE +
                dht11->num_edges - DHT11_EDGES_PER_READ;
        for (; offset >= 0; --offset) {
            ret = dht11_decode(dht11, offset);
            if (!ret)
                break;
        }

        if (ret)
            goto err;
    }

    ret = IIO_VAL_INT;
    if (chan->type == IIO_TEMP)
        *val = dht11->temperature;
    else if (chan->type == IIO_HUMIDITYRELATIVE)
        *val = dht11->humidity;
    else
        ret = -EINVAL;
err:
    dht11->num_edges = -1;
    mutex_unlock(&dht11->lock);
    return ret;
}
```

Read Raw Data

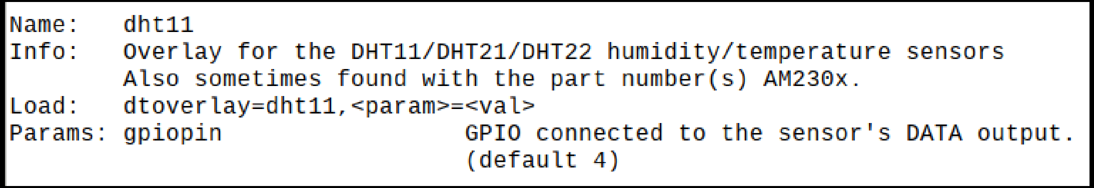
Summary of iio driver program

- 1. Define, initialize & register
  - `iio_device_alloc()` → initialize → `iio_device_register();` // register platform_device
  - `struct platform_driver dht11_driver{};`// define platform_driver
  - `module_platform_driver(dht11_driver);` // register platform_driver →__platform_driver_register()
- 2. Set up channel
  - `struct iio_chan_spec dht11_chan_spec[]`
- 3. implement read / write
  - `dht11_read_raw()`
- 4. `iio_info`
  - Put `dht11_read_raw()` in `struct iio_info dht11_iio_info`
- 5. implement and register `iio_dev`

Appendix A
dht11 implementation



1. Pre-requirement
   1. Pinout connection
   2. check /boot/overlay/README



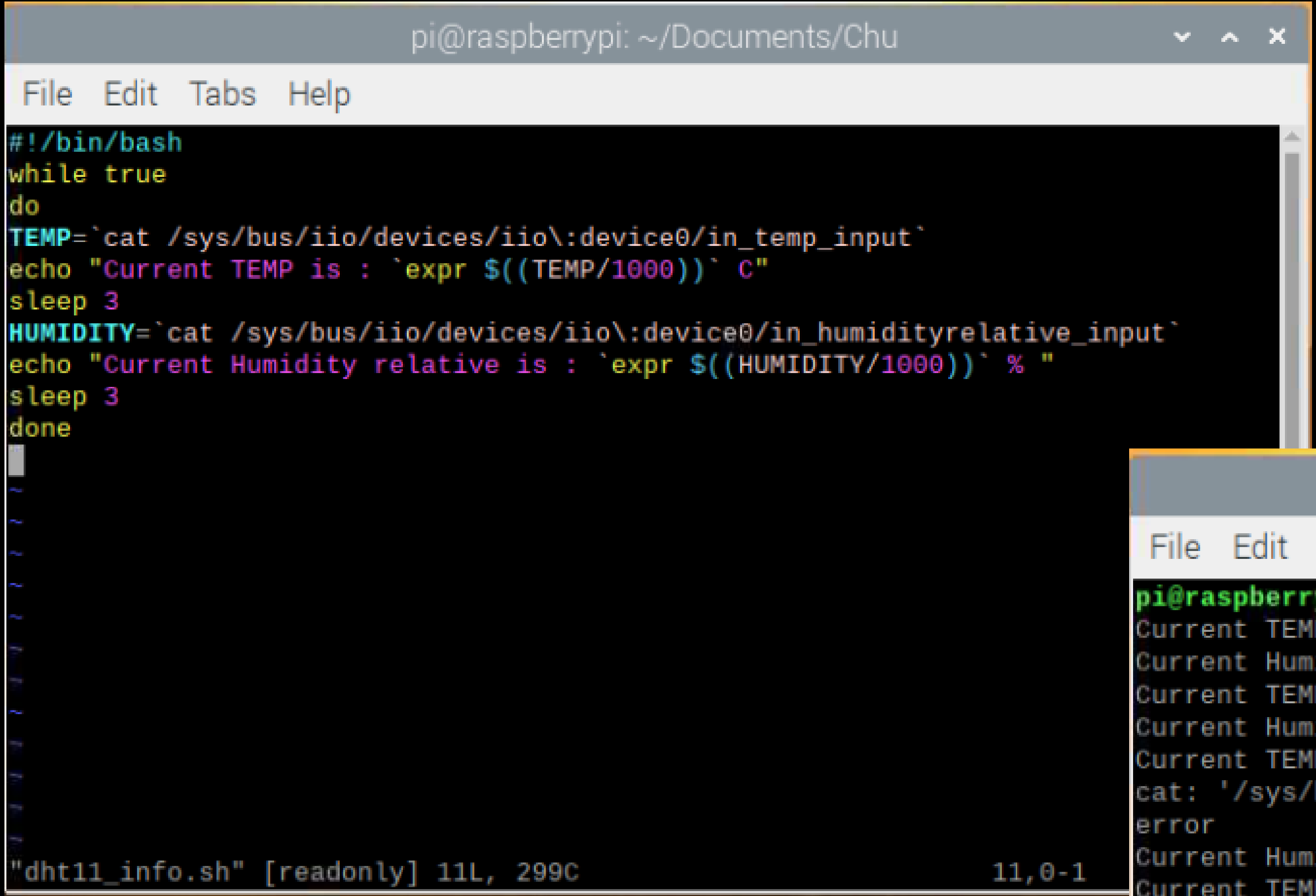
```
Name:   dht11
Info:   Overlay for the DHT11/DHT21/DHT22 humidity/temperature sensors
        Also sometimes found with the part number(s) AM230x.
Load:   dtoverlay=dht11,<param>=<val>
Params: gpiopin                 GPIO connected to the sensor's DATA output.
                                (default 4)
```

1. Modify Configuration
   1. /boot/config.txt

```
sudo su –
vim /boot/config.txt
device_tree = bcm2710-rpi-3-b.dtb
dtparam = i2c_arm = on
dtoverlay=dht11
```

2. Write Shell Script code for dht11

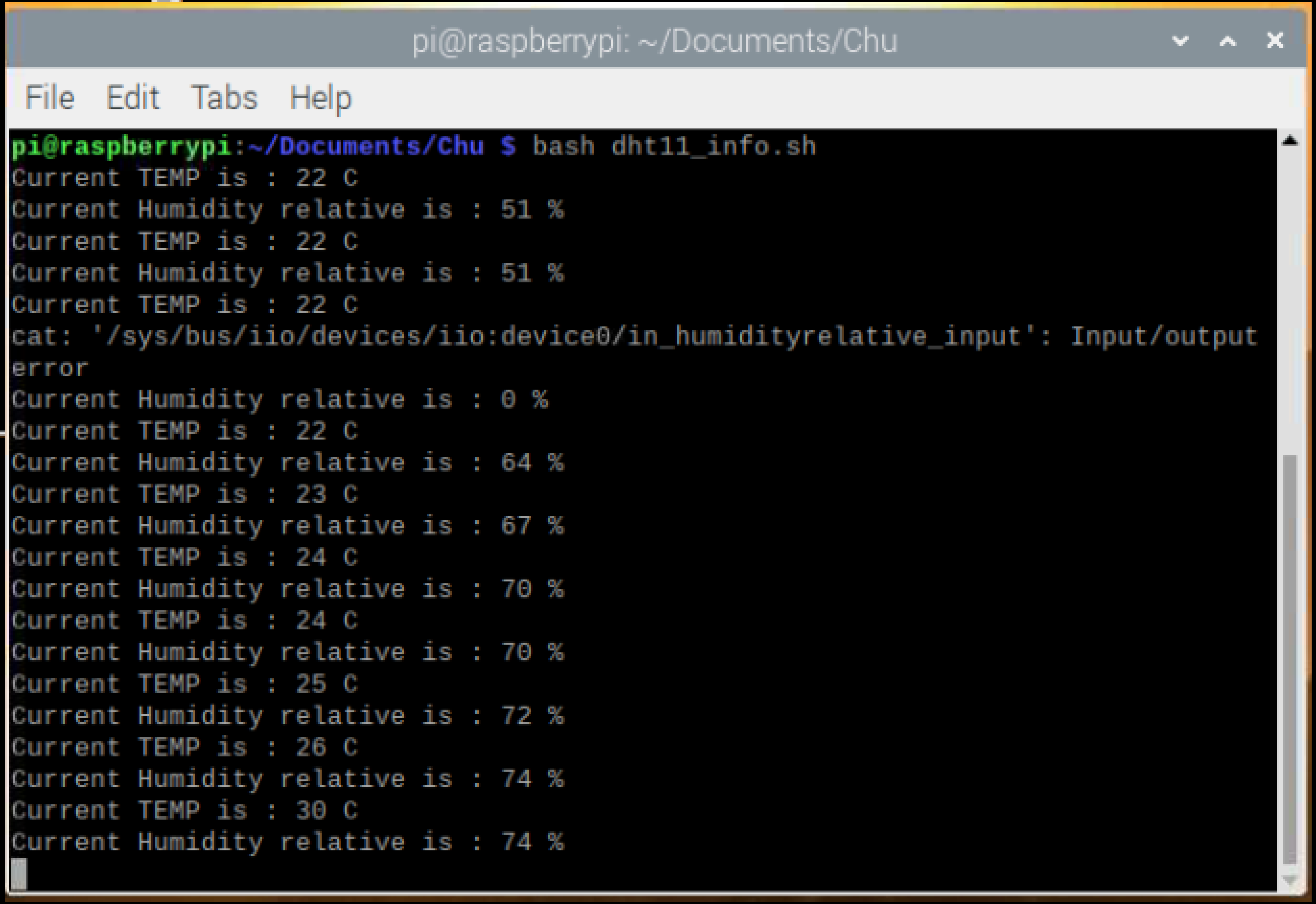
3. run code



pi@raspberrypi: ~/Documents/Chu

File Edit Tabs Help

```
#!/bin/bash
while true
do
TEMP=`cat /sys/bus/iio/devices/iio\:device0/in_temp_input`
echo "Current TEMP is : `expr $((TEMP/1000))` C"
sleep 3
HUMIDITY=`cat /sys/bus/iio/devices/iio\:device0/in_humidityrelative_input`
echo "Current Humidity relative is : `expr $((HUMIDITY/1000))` % "
sleep 3
done

~
~
~
~
~
~
~
~
~
~
~
~
"dht11_info.sh" [readonly] 11L, 299C                           11,0-1
```



pi@raspberrypi: ~/Documents/Chu

File Edit Tabs Help

```
pi@raspberrypi:~/Documents/Chu $ bash dht11_info.sh
Current TEMP is : 22 C
Current Humidity relative is : 51 %
Current TEMP is : 22 C
Current Humidity relative is : 51 %
Current TEMP is : 22 C
cat: '/sys/bus/iio/devices/iio:device0/in_humidityrelative_input': Input/output
error
Current Humidity relative is : 0 %
Current TEMP is : 22 C
Current Humidity relative is : 64 %
Current TEMP is : 23 C
Current Humidity relative is : 67 %
Current TEMP is : 24 C
Current Humidity relative is : 70 %
Current TEMP is : 24 C
Current Humidity relative is : 70 %
Current TEMP is : 25 C
Current Humidity relative is : 72 %
Current TEMP is : 26 C
Current Humidity relative is : 74 %
Current TEMP is : 30 C
Current Humidity relative is : 74 %
```