# Internet of Things

IoT Drivers allow any Odoo module to communicate in real-time with any device connected to the IoT Box. Communication with the IoT Box goes both ways, so the Odoo client can send commands to and receive information from any of the supported devices.
To add support for a device, all we need is:

- an **`Interface`**, to detect connected devices of a specific type
- a **`Driver`**, to communicate with an individual device

At each boot, the IoT Box will load all of the Interfaces and Drivers that can be located on the connected Odoo instance. Each module can contain an **`iot_handlers`** directory that will be copied to the IoT Box. The structure of this directory is the following

```
your_module
├── ...
└── iot_handlers
    ├── drivers
    |   ├── DriverName.py
    |   └── ...
    |
    └── interfaces
        ├── InterfaceName.py
        └── ...
```

## Detect Devices

Devices connected to the IoT Box are detected through **`Interfaces`**. There is an Interface for each supported connection type (USB, Bluetooth, Video, Printers, Serial, etc.). The interface maintains a list of detected devices and associates them with the right Driver.
Supported devices will appear both on the IoT Box Homepage that you can access through its IP address and in the IoT module of the connected Odoo instance.

### Interface

The role of the Interface is to maintain a list of devices connected through a determined connection type. Creating a new interface requires

- Extending the **`Interface`** class
- Setting the **`connection_type`** class attribute
- Implementing the **`get_devices`** method, that should return a dictionary containing data about each detected device. This data will be given as argument to the constructors and

`supported` method of the Drivers.

> Setting the `_loop_delay` attribute will modify the interval between calls to `get_devices` . By default, this interval is set to 3 seconds.

```
from odoo.addons.hw_drivers.interface import Interface

class InterfaceName(Interface):
    connection_type = 'ConnectionType'

    def get_devices(self):
        return {
            'device_identifier_1': {...},
            ...
        }
```

## Driver

Once the interface has retrieved the list of detected devices, it will loop through all of the Drivers that have the same `connection_type` attribute and test their respective `supported` method on all detected devices. If the supported method of a Driver returns `True` , an instance of this Driver will be created for the corresponding device.

> `supported` methods of drivers are given a priority order. The `supported` method of a child class will always be tested before the one of its parent. This priority can be adjusted by modifying the `priority` attribute of the Driver.

Creating a new Driver requires:

- Extending `Driver`
- Setting the `connection_type` class attribute.
- Setting the `device_type` , `device_connection` and `device_name` attributes.
- Defining the `supported` method

```
from odoo.addons.hw_drivers.driver import Driver

class DriverName(Driver):
    connection_type = 'ConnectionType'

    def __init__(self, identifier, device):
        super(NewDriver, self).__init__(identifier, device)
        self.device_type = 'DeviceType'
        self.device_connection = 'DeviceConnection'
        self.device_name = 'DeviceName'

    @classmethod
    def supported(cls, device):
        ...
```

# Communicate With Devices

Once your new device is detected and appears in the IoT module, the next step is to communicate with it. Since the box only has a local IP address, it can only be reached from the same local network. Communication, therefore, needs to happen on the browser-side, in JavaScript.

The process depends on the direction of the communication: - From the browser to the box, through Actions - From the box to the browser, through Longpolling

Both channels are accessed from the same JS object, the **`DeviceProxy`**, which is instantiated using the IP of the IoT Box and the device identifier.

```
var DeviceProxy = require('iot.DeviceProxy');

var iot_device = new DeviceProxy({
    iot_ip: iot_ip,
    identifier: device_identifier
});
```

## Actions

Actions are used to tell a selected device to execute a specific action, such as taking a picture, printing a receipt, etc.

> It must be noted that no “answer” will be sent by the box on this route, only the request status. The answer to the action, if any, has to be retrieved via the longpolling.

An action can be performed on the DeviceProxy Object.

```
iot_device.action(data);
```

In your driver, define an **`action`** method that will be executed when called from an Odoo module. It takes the data given during the call as argument.

```
def action(self, data):
    ...
```

## Longpolling

When any module in Odoo wants to read data from a specific device, it creates a listener identified by the IP/domain of the box and the device identifier and passes it a callback function to be called every time the device status changes. The callback is called with the new data as argument.

```
iot_device.add_listener(this._onValueChange.bind(this));

_onValueChange: function (result) {
    ...
}
```

In the Driver, an event is released by calling the `device_changed` function from the `event_manager`. All callbacks set on the listener will then be called with `self.data` as argument.

```
from odoo.addons.hw_drivers.event_manager import event_manager

class DriverName(Driver):
    connection_type = 'ConnectionType'

    def methodName(self):
        self.data = {
            'value': 0.5,
            ...
        }
        event_manager.device_changed(self)
```