# Transceiver Concept Overview

**Applicative Functionalities**     **Platform Functionalities**     **PHY**

**Upper Layers**

…

*Modem*     *Transceiver*     *Antenna*     *Air interface*
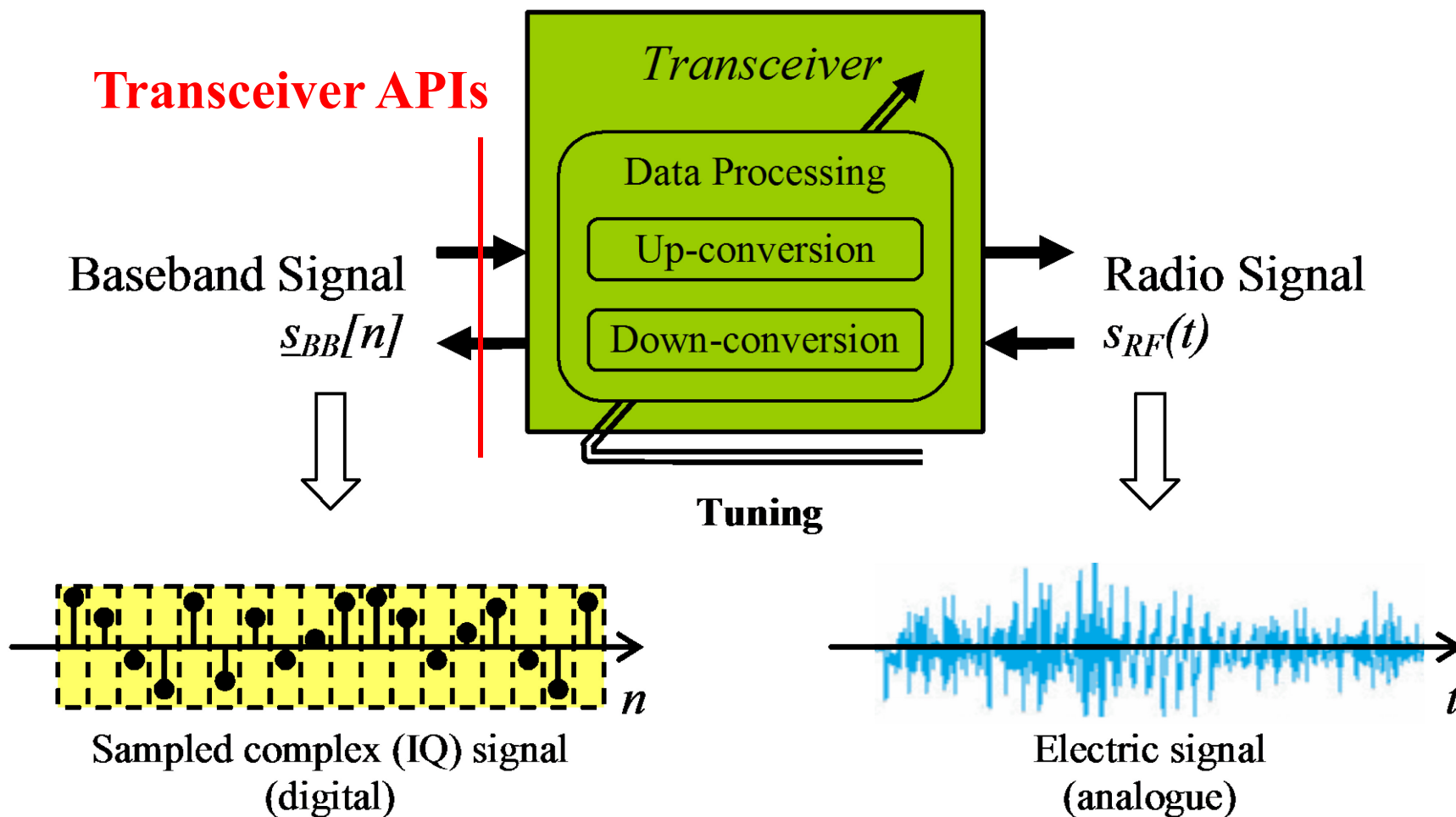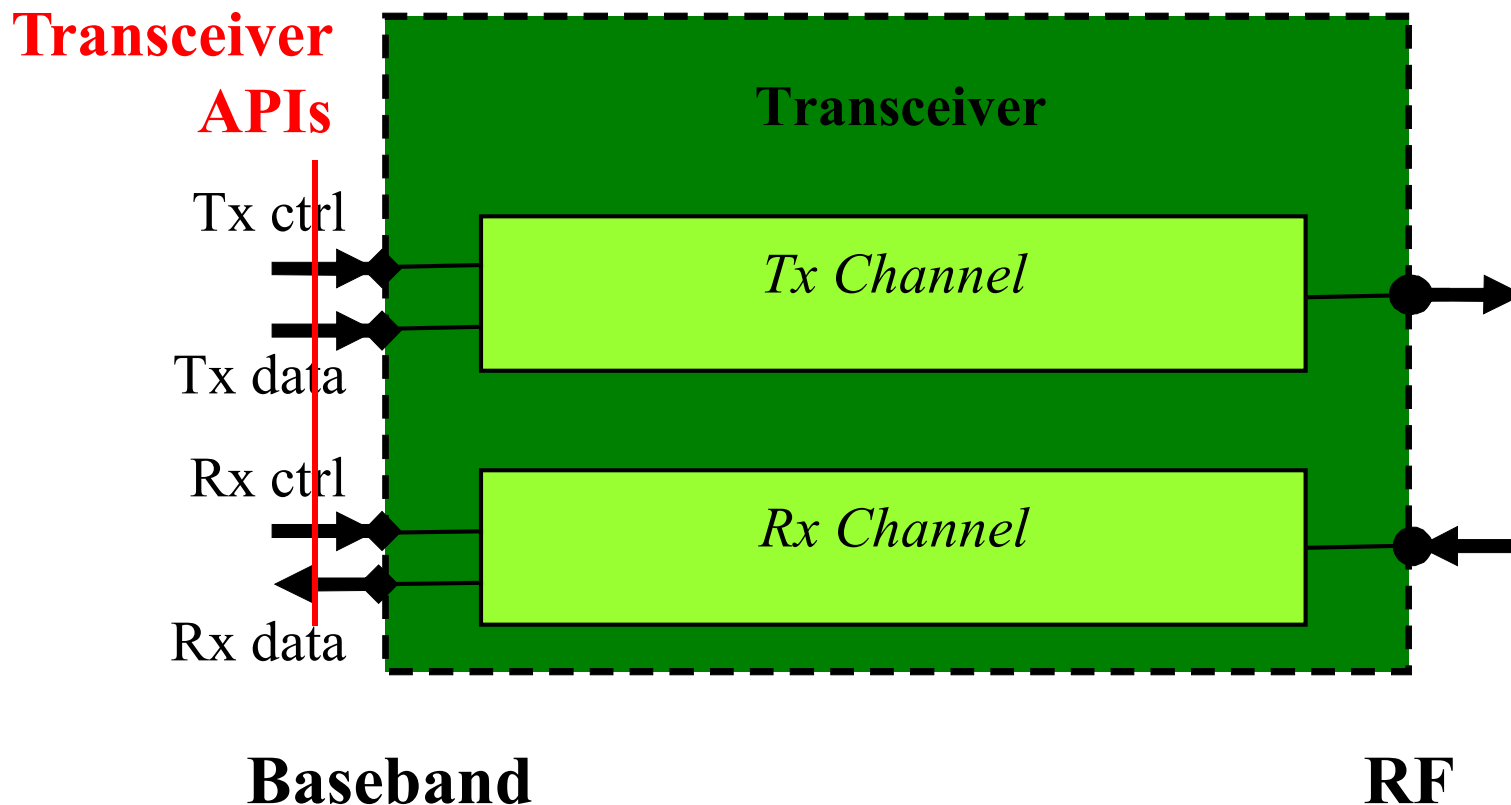
## Transceiver API

The Transceiver Interface API is a generic API aiming at becoming an standardized API for SDR architectures (developed within the Wireless Innovation Forum)

It is a functional specification for RF Hardware platforms command and control. It sits within the physical layer (L1).

**Transceiver APIs**

*Transceiver*

Data Processing

Up-conversion

Down-conversion

Baseband Signal
$\underline{s}_{BB}[n]$

Radio Signal
$s_{RF}(t)$

**Tuning**

Sampled complex (IQ) signal
(digital)

$n$

Electric signal
(analogue)

$t$

# Tx and Rx Channels



- **Tx and Rx channels with dedicated Data and Control interfaces**
- **Common description for Half Duplex, Full Duplex & Simplex**
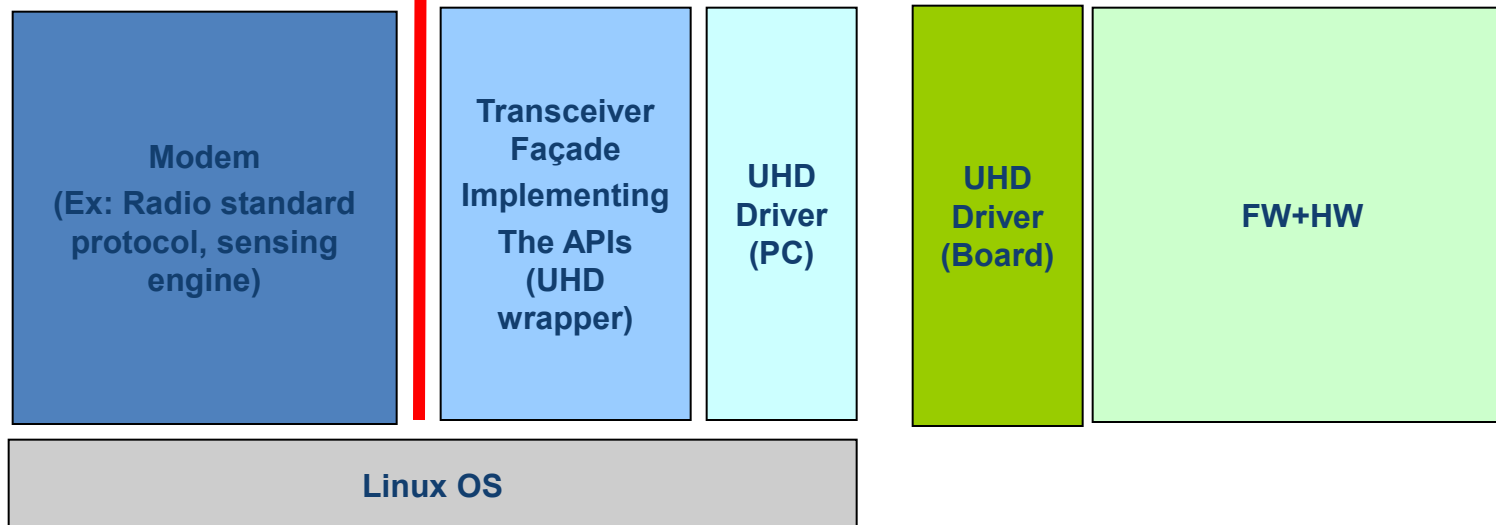- **No explicit direction setting interface is required**

# Implementation available for CREW

Linux PC

USRP2

API operations

| Modem (Ex: Radio standard protocol, sensing engine) | Transceiver Façade Implementing The APIs (UHD wrapper) | UHD Driver (PC) | UHD Driver (Board) | FW+HW |

Linux OS

```
createTransmitCycleProfile(
    TimeRequest requestedTransmitStartTime,
    TimeRequest requestedTransmitStopTime,
    UShort requestedTuningPreset,
    Frequency requestedCarrierFrequency,
    AnaloguePower requestedNominalRFPower);


configureTransmitCycle(
    Ulong targetCycleId,
    TimeRequest requestedTransmitStartTime,
    TimeRequest requestedTransmitStopTime,
    Frequency requestedCarrierFrequency,
    AnaloguePower requestedNominalRFPower);


setTransmitStopTime(
    Ulong                 targetCycleId,
    TimeRequest           requestedTransmitStopTime);


pushBBSamplesTx(
    BBSamplesPacket * thePushedPacket,
    Boolean                    endOfBurst);
```

```
createReceiveCycleProfile(
    TimeRequest requestedReceiveStartTime,
    TimeRequest requestedReceiveStopTime,
    UShort requestedTuningPreset,
    ULong requestePacketSize,
    Frequency requestedCarrierFrequency);


configureReceiveCycle(
    Ulong targetCycleId,
    TimeRequest requestedReceiveStartTime,
    TimeRequest requestedReceiveStopTime,
    ULong requestePacketSize,
    Frequency requestedCarrierFrequency);


setReceiveStopTime(
    Ulong              targetCycleId,
    TimeRequest        requestedReceiveStopTime);


pushBBSamplesRx(
    BBSamplesPacket * thePushedPacket,
    Boolean endOfBurst);
```