So far we have (partial) support to switch off the power to a physical BTS / rf transceiver hardware.

What we don't yet have is support to power down the osmo-trx machine when it's not needed.

In the osmo-gsm-tester prod setup, it is possible to switch power to that machine via intellinet at 10.42.42.250 port 6.

I'm not sufficiently familiar with the code to judge if we can just treat it like power-switching the actual RF hardware (B200, limesdr, ...) or not.

First add the power_supply node with 10.42.42.250 port 6 to here:

```yaml
- label: Ettus B200
type: osmo-bts-trx
ipa_unit_id: 6
addr: 10.42.42.50
band: GSM-1800
trx_list:
  - arfcn: 850
  - arfcn: 852
ciphers: [a5_0, a5_1, a5_3]
osmo_trx:
  type: uhd
  launch_trx: true
  remote_user: jenkins
  trx_ip: 10.42.42.116
dev_args: "type=b200,serial=306BD11"
clock_reference: external
multi_arfcn: true
```

Like it's currently done for SCSK:

```yaml
- label: sysmoCell 5000
type: osmo-bts-trx
ipa_unit_id: 7
addr: 10.42.42.51
band: GSM-1800
ciphers: [a5_0, a5_1, a5_3]
trx_list:
  - power_supply:
      type: 'intellinet'
      device: '10.42.42.250'
      port: '5'
arfcn: 862
```

Then bts_osmotrx.py should already power cycle the mentioned slot during start() if it was configured.

What we are missing is extending OsmoTrxUHD and OsmoTrxLMS like it's already done in TrxSC5, which basically means trying to connect to the remote host and running something there ("ls" or whatever) from time to time until the host is up and succeeds, and then trx_ready() should return true.
Quick diff of what should be needed, for illustration:

diff --git a/src/osmo_gsm_tester/obj/bts_osmotrx.py b/src/osmo_gsm_tester/obj/bts_osmotrx.py
index 8eed8cf..181d668 100644
--- a/src/osmo_gsm_tester/obj/bts_osmotrx.py
+++ b/src/osmo_gsm_tester/obj/bts_osmotrx.py
@@ -278,12 +278,30 @@ class OsmoTrx(Trx, metaclass=ABCMeta):
     proc.launch()
     return proc

+    def wait_remote_is_up(self)
+        name = "true"
+        run_dir = self.run_dir.new_dir(name)
+        popen_args = ('true')
+        proc = process.RemoteProcess(name, run_dir, self.remote_user, self.listen_ip, None,
+                                popen_args)
+        keepTrying = 10
+        while keepTrying > 0:
+            if proc.respawn_sync(raise_nonsuccess=False) == 0):
+                break
+                keepTrying = keepTrying - 1
+                self.log('Accessing osmo-trx remote failed, retrying %d more times' % keepTrying)
+                MainLoop.sleep(5)
+            if keepTrying == 0:
+                raise log.Error('Failed configuring osmo-trx-remote!')

-    def start_remotely(self, keepalive):
-        # Run remotely through ssh. We need to run osmo-trx under a wrapper
-        # script since osmo-trx ignores SIGHUP and will keep running after
-        # we close local ssh session. The wrapper script catches SIGHUP and
-        # sends SIGINT to it.
+    def start_remotely(self, keepalive):
+        # Run remotely through ssh. We need to run osmo-trx under a wrapper
+        # script since osmo-trx ignores SIGHUP and will keep running after
+        # we close local ssh session. The wrapper script catches SIGHUP and
+        # sends SIGINT to it.
+        wait_remote_is_up()
+        rem_host = remote.RemoteHost(self.run_dir, self.remote_user, self.listen_ip)
+        remote_prefix_dir = util.Dir(OsmoTrx.REMOTE_DIR)