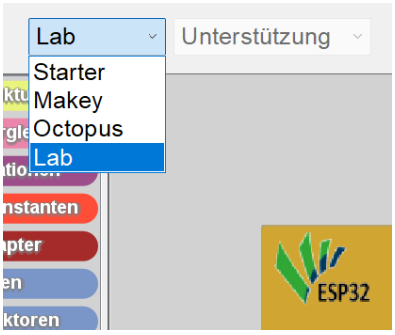
Was ist nötig, um eine eigene Hardware in die Werkstatt zu integrieren?

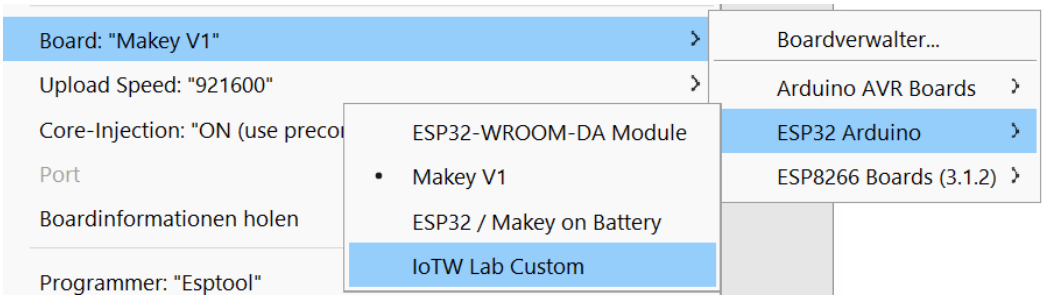
Hier am Beispiel des „Custom Lab“-Boards:



1. Anlegen eines neuen Boards in der Datei Boards.txt

Die Datei befindet sich unter
\IoTW\arduino\portable\packages\esp32\hardware\esp32\3.3.2

In dieser Textdatei verwaltet die IDE 1 alle unter dem Menüpunkt Werkzeuge angezeigten Boards. Um den Einstieg zu erleichtern, sind im Setup der IoT-Werkstatt die meisten Boards durch den Eintrag .hide=true versteckt und werden nicht angezeigt.



Wir erstellen ein neues Board durch Hinzufügen der Boardbeschreibung:

```
####################
IoTW_Lab32.name=IoTW Lab Custom
IoTW_Lab32.bootloader.tool=esptool_py
IoTW_Lab32.bootloader.tool.default=esptool_py
IoTW_Lab32.upload.tool=esptool_py
IoTW_Lab32.upload.tool.default=esptool_py
IoTW_Lab32.upload.tool.network=esp_ota
IoTW_Lab32.upload.maximum_size=1310720
IoTW_Lab32.upload.maximum_data_size=327680
IoTW_Lab32.upload.flags=
IoTW_Lab32.upload.extra_flags=
IoTW_Lab32.serial.disableDTR=true
IoTW_Lab32.serial.disableRTS=true
IoTW_Lab32.build.tarch=xtensa
```

```
IoTW_Lab32.build.bootloader_addr=0x1000
IoTW_Lab32.build.target=esp32
IoTW_Lab32.build.mcu=esp32
IoTW_Lab32.build.core=esp32_pre
IoTW_Lab32.build.variant=makey_v1
IoTW_Lab32.build.board=D1_MINI32
IoTW_Lab32.build.f_cpu=240000000L
IoTW_Lab32.build.flash_mode=dio
IoTW_Lab32.build.flash_size=4MB
IoTW_Lab32.build.boot=dio
IoTW_Lab32.build.partitions=default
IoTW_Lab32.build.defines=
```

Wichtig ist vor allem der Menüeintrag zur Aktivierung des precompilierten esp32 Cores. Dieser beschleunigt den späteren Übersetzungsvorgang mit dem C-Compiler:

```
# Inject-Menü (schaltet Core + Flag)
IoTW_Lab32.menu.inject.on=ON (use precompiled core.a)
IoTW_Lab32.menu.inject.on.build.core=esp32_pre
IoTW_Lab32.menu.inject.on.build.inject_core=1

IoTW_Lab32.menu.inject.off=OFF (compile full core)
IoTW_Lab32.menu.inject.off.build.core=esp32
IoTW_Lab32.menu.inject.off.build.inject_core=0

IoTW_Lab32.menu.UploadSpeed.921600=921600
IoTW_Lab32.menu.UploadSpeed.921600.upload.speed=921600
IoTW_Lab32.menu.UploadSpeed.115200=115200
IoTW_Lab32.menu.UploadSpeed.115200.upload.speed=115200
IoTW_Lab32.menu.UploadSpeed.460800.linux=460800
IoTW_Lab32.menu.UploadSpeed.460800.macosx=460800
IoTW_Lab32.menu.UploadSpeed.460800.upload.speed=460800
IoTW_Lab32.menu.UploadSpeed.1500000=1500000
IoTW_Lab32.menu.UploadSpeed.1500000.upload.speed=1500000
IoTW_Lab32.upload.erase_cmd=
##############################################################
```

Damit haben wir ein `IoTW_Lab32` Board angelegt. In der IDE erscheint dieses Board mit dem Namen „IoTW Lab Custom“ im Auswahlfeld.

2. Pinbelegung definieren

Jetzt legen wir die GPIO Pins fest, die der Codegenerator bei der Erstellung des C-Codes verwendet. Dazu erweitern wir die Definition arduino_pins.h im Varianten-verzeichnis unter:

\IoTW\arduino\portable\packages\esp32\hardware\esp32\3.3.2\variants\IoTW_Lab32

```
// IoT-Werkstatt
#define IOTW_BOARD_MAKEY
#define IOTW_BOARD_MAKEY_V1

#define IOTW_GPIO_ROTARY_B 35
#define IOTW_GPIO_ROTARY_A 32
```

```
#define IOTW_GPIO_ROTARY_BUTTON 15
#define IOTW_GPIO_NEO 13
#define IOTW_GPIO_NEOWING 33
#define IOTW_GPIO_A0 A5
#define IOTW_AI_SCALE 1
#define IOTW_GPIO_AUDIO_OUT 25

// Number of maximal possible MQTT Subscriptions
#define IOTW_MAX_MQTT_SUB 10

// Arraylänge beim Datenfeld (z.B. OLED Array display)
  #define IOTW_ARRAYLEN 64
// Feldlänge bei EdgeImpulse (maximal mögliche Einträge für features-
Vector)
  #define IOTW_EI_MAXPOINTS 50

// Mikrofon
#define IOTW_GPIO_MIC_BCK      GPIO_NUM_12   // Bit-Clock
#define IOTW_GPIO_MIC_WS       GPIO_NUM_2    // LR-Clock
#define IOTW_GPIO_MIC_SD       GPIO_NUM_27   // Daten-IN

// LoRaWAN radio MCCI LMIC über Feather
// #define CFG_sx1262_radio 1
#define CFG_sx1276_radio 1
#define IOTW_GPIO_LMIC_NSS 14
#define IOTW_GPIO_LMIC_DIO0 33
#define IOTW_GPIO_LMIC_DIO1 33
#define IOTW_GPIO_LMIC_RST 255
```