AxDinoESP32 API

Version 2.0
5-14-2021

Introduction

The AxDinoESP32 Arduino library provides an interface to the hardware controlled by the ESP32 and implements the following functionality.

1. Provides a simple method-based API to control the hardware from an Arduino sketch.
2. Automatically reads the Wheelchair Relay box inputs and optionally switches the output relays enabling wheelchair control.
3. Automatically reads the temperature sensor and optionally switches on the fan when the temperature exceeds a threshold and switches off the fan when the temperature falls 5°C below the threshold.

Installation

Install the AxDinoESP32 library directory in your Arduino environment local libraries directory.

You also need to install the following additional libraries.

1. KMPProDinoESP32 - download from https://kmpelectronics.eu/tutorials-examples/prodino-esp32-versions-tutorial/
2. OneWire - May be installed using the Arduino Library Manager
3. DallasTemperature - May be installed using the Arduino Library Manager

Use

Assuming ESP32 support as been added to the Arduino environment, set the Board type to “ESP32 Dev Board” and the serial Port to the port connected to the ESP32.

Include the library in the Arduino sketch.

```
#include “AxDinoESP32.h”
```

It is not necessary to include any other other libraries. They are included automatically.

Initialize the Serial output and the library in the `setup` routine. The Serial output is necessary for diagnostic logging.

```
Serial.begin(115200);
AxDinoESP32.begin();
```

The AxDinoESP32 class is automatically instanced by the library.

The “lib_test” demo shows basic operation and can be found in the library `examples` sub-directory.

API

`void begin()` - Initialize the controller and start temperature and input monitoring and automatic fan and output relay control (both enabled by default). Must be called before any other API method.

Inputs - none
Return - none

`char* getVersion()` - Return a pointer to a character string containing the current library version, e.g. “1.0”.

Inputs - none
Return - pointer to constant char string

`float getTemp(int units)` - Return the current temperature sensor value in the specified units (C or F).

Inputs

units - One of the constants AD_TEMP_C or AD_TEMP_F
Return - Floating point temperature value in the specified units

`float getFanAutoTemp()` - Return the automatic fan control temperature threshold (C).
Inputs - none
Return - Temperature threshold in degrees C

`void setFanAutoTemp(float t, int units)` - Specify the temperature threshold for automatic fan control.
Inputs
t - Floating point temperature value interpreted in the specified units
units - One of the constants AD_TEMP_C or AD_TEMP_F
Return - none

Note: The default temperature threshold is 40°C.

`bool getSwitch(int n)` - Return the current Wheelchair Relay box switch inputs used for automatic output enables.
Inputs
n - An integer value in the range 0 - 3 specifying the following Wheelchair relay controls (“Forward”, “Reverse”, “Right”, “Left”). These switch values are mapped to the four relay outputs.
Return - true if the switch is set, false if the switch is clear.

`void setOutput(int n, bool state)` - Manually control one of the outputs (Pi power enable, Fan power enable and four relay outputs). Note that clearing the Pi output may have no effect if the Wheelchair relay output associated with it (“Horn”) is set, clearing the Fan output may have no effect if automatic fan functionality has been enabled and clearing a relay may have no effect if the automatic switch functionality has been enabled then (over-ridden by Wheelchair relay controls).
Inputs
n - One of the constants AD_OUT_PI, AD_OUT_FAN, AD_OUT_RLY_1, AD_OUT_RLY_2, AD_OUT_RLY_3, AD_OUT_RLY_4
state - Set true to enable an output, false to clear an output
Return - none

`bool getOutput(int n)` - Return the current output enable value based on manual control via the setOutput method and automatic mechanisms if enabled.
Inputs
n - One of the constants AD_OUT_PI, AD_OUT_FAN, AD_OUT_RLY_1, AD_OUT_RLY_2, AD_OUT_RLY_3, AD_OUT_RLY_4
Return - true if the output is currently enabled, false if it is not

Note: The Pi output only monitors the manual control as the Wheelchair relay control signal bypasses the ESP32 so that the Pi power can be maintained while the ESP32 is reprogrammed.

`bool getAutoOutput(int n)` - Return the status of an output channel based on the Relay Box inputs.
Inputs
n - One of the constants AD_OUT_PI, AD_OUT_FAN, AD_OUT_RLY_1, AD_OUT_RLY_2, AD_OUT_RLY_3, AD_OUT_RLY_4
Return - true if the output is currently enabled, false if it is not

Note: The Pi output only monitors the manual control as the Wheelchair relay control signal bypasses the ESP32 so that the Pi power can be maintained while the ESP32 is reprogrammed.

`bool getAutoFan()` - Get state of automatic control of fan.
Inputs - none
Return - true if the fan is enabled due to automatic control.

`void enableAutoFan(bool state)` - Control the automatic fan control mechanism. When enabled the fan is automatically switched on when the temperature exceeds the threshold.
Inputs
state - Set true to enable automatic fan control, false to disable automatic fan control.
Return - none

Note: Automatic fan control is enabled by default.

`bool getAutoSwitch()` - Get automatic control of the four relays state.
Inputs - none
Return - true if automatic control of the relays is enabled

`void enableAutoSwitch(bool state)` - Control the automatic switch functionality. When enabled the four relay outputs are automatically controlled by the Wheelchair relay control signals.
Inputs
state - Set true to enable automatic switch functionality, false to disable automatic switch functionality.
Return - none

Note: Automatic switch functionality is enabled by default.

`void setLED(RgbColor color)` - Set the RGB LED on the KMP PRODino PCB to the specified color.
Inputs
color - LED color as specified by the RgbColor type in the KMPProDinoESP32 library.
Return - none