OCTOPUS

Schnellstart in die Welt des Internet der Ding mit dem #IoT Octopus:

Ideen und Blaupause (extern DIV, PDF)

Ardublock für den #IoT Octopus including QuickStart Guide (extern, UCB, ZIP, >500MB)

Hintergrund zur **IoT Werkstatt**:

> *Pressemitteilung: IoT-Werkstatt bringt das Internet der Dinge spielerisch in die Bildungssysteme*

https://www.umwelt-campus.de/ucb/index.php?id=iot-werkstatt

The **OCTOPUS IoT Badge** *limited developer* edition – getting started:

First congratulations to your all-in-one education and innovation kit for your Internet of Things #IoT applications!

The kit is based in its core around the famous ESP8266 (SoC) from Esperiff and uses the latest ESP8266-12F module. More details you can find while checking the schematic. As the hardware is open – we like you to have a look into details, and make your modifications while referencing to the www.fab-lab.eu

source. *If your like the boards and need any custom design e.g. your company/ lab/organization logo … or a different PCB color – please let us know!*

Please be aware this kit is for education and lab use only. As it is made by Maker for Maker please let us know about any add ons or improvements you might find out. The section below might include external links which we are not maintaining nor are responsible for. Code samples are as they are, might include third party work and resources.

(why to we need all these disclosures!?!? anyway we trust you that you are not going to burn your fingers!) **– Now let's get it running!**

Frist you do need to decide what power source you might want to use:

a. ready-to-run: USB and an external 3xAAA or 3xAA battery pack, **no** modification needed
-> ***just start, have fun!*** *There might be code already on the board if the right NeoPixel is green,*
you are lucky, breath on to the sensor (BME280) and it will detect high humidity! Make sure you got the USB driver and Arduino IDE running:
https://learn.adafruit.com/adafruit-feather-huzzah-esp8266/using-arduino-ide
http://www.silabs.com/products/mcu/pages/usbtouartbridgevcpdrivers.aspx

b. professionals only: LiPo with charging via USB or solar panel
c. professionals only: use only 2xAAA batteries on the board, this is a hack!

Options b. and c. are requiring modifications and can ***NOT*** *be used in parallel, nevertheless you might change during the time and reverse the modifications.*

Safety note: if you are going to use a LiPo please make sure you understand the safety instructions handling LiPo batteries, specially while charing always keep an eye on it, don't short cut or overheat LiPo batteries. Work on your own risk! We suggest only Pros to be using the LiPo option.

When completed you are ready to run the OCTOPUS. Plug it to your computers USB port – the port is protected by an fuse max. 500mA and a reverse power diode. If you are using the 2xAAA hack – there is no reverse power protection – the USB power is not used, so make sure you always follow the printed battery orientation while insertion the batteries.

Pinout:

Now have fun!

All you do need to deploy your first application, follow these great tutorials – and
if you like them consider Adafruit as a source for your supplies too! (we love them)

https://learn.adafruit.com/adafruit-feather-huzzah-esp8266/overview
https://learn.adafruit.com/adafruit-feather-huzzah-esp8266/using-arduino-ide
https://learn.adafruit.com/adafruit-oled-featherwing/overview
https://learn.adafruit.com/adafruit-15×7-7×15-charlieplex-led-matrix-charliewing-featherwing
https://learn.adafruit.com/adafruit-neopixel-featherwing/overview
https://learn.adafruit.com/adafruit-neopixel-uberguide/overview
https://learn.adafruit.com/adafruit-bme280-humidity-barometric-pressure-temperature-sensor-breakout/wiring-and-test

MacOS might not recognize the USB/UART driver CP210x you can manually install:
http://www.silabs.com/products/mcu/pages/usbtouartbridgevcpdrivers.aspx

Btw there is a hackathon, which is going to use OCTOPUS with some more instructions on coding and Arduino(TM) IDE examples:

Octopus ArduBlock Samples / Instructions
Octopus Arduino(TM) Sampes / Instructions

Octopus – SDK download Zip, 555MB(!) incl. schematic, instructional videos

If you are interested in a ultra-low-power hack get more background here:

https://www.hackster.io/fablabeu/esp8266-thing-by-sparkfun-982bc6

A excellent introduction to the Octopus by Carl-Benz-School in Koblenz
in this video tutorial:

link was deleted – due to yahoo video source …

Our list of tested add on components:

Useful parts:
https://www.adafruit.com/products/2975
https://www.adafruit.com/products/2884 (breakout)
https://www.adafruit.com/products/790 (GPS, requires virtual serial for ESP8266!)

https://www.adafruit.com/products/2940 (using the short header for a **slim design**)
https://www.adafruit.com/products/3002 (using the short header for a **slim design**)

Displays:
https://www.adafruit.com/products/2965 (LED Matrix Display)
https://www.adafruit.com/products/2900 (OLED)

https://www.adafruit.com/products/2945 (NeoPixel Matrix, read modification for ESP8266, found here: https://learn.adafruit.com/adafruit-neopixel-featherwing/pinouts #16 pin!)
https://www.adafruit.com/products/3108 (7-Segment, for a watch face or digital multi meter)
https://www.adafruit.com/products/3130 (Alpha-Segment)
https://www.adafruit.com/products/3315 (TFT / Touch)

Controls:
https://www.adafruit.com/products/2928 (PWM 8x)
https://www.adafruit.com/products/2927 (Motor 4x / Stepper 2x)

https://www.adafruit.com/products/3231 (LoRa)

SeeedStudio Grove:
AirQuality, Dust Sensor, Speech Recognizer (virtual serial!), … most analog or I2C sensor with support for 3V (!)

FAQ:

- using NeoPixel with RGBW support, that means there is a true „white“, make sure you are setting RGBW when initializing it (see adafruit NeoPixel Uberguide)
- you can run NeoPixel (as we do on the PCB) with 3.3V, even it is out of spec it works, if you add external stripes make sure you monitor the current(!) – THIS is a hack but works only limit we do see is the strength of the blue LED (as it forward voltage is close to 3V)

a. Ready-to-run setup – USB power and external Batterie support (e.g. 3xAA, 3xAAA)

Parts you got with the kit

Solder Rotary Encoder

Solder Headers – we recommend the short headers, for a flat design

Solder Groove Headers

b. Enable LiPo Option: Please be sure you understand how to safely Charge LiPos (see tutorials of Sparkfun or Adafruit). Never leave the unit unattended during charging. Use only LiPos with build in protection circuit (current, voltage). Based on the current configuration use at least a

capacity of >=350mAh. See schematics for details. *Tip: you can attach a slider switch S1 to switch of the LiPo: before remove R22 and D2*

Solder 2-pin JST (SMD) *not included in the kit (polarity is marked with a „+“ on the PCB)*

Close this solder jumper – enabling charging *(if you are lucky there is a Zero-Ohm resistor in the kit,*
otherwise just use a short wire to bridge the two pads)

If you do want to charge the LiPo via a solar panel you can use a micro USB plug to connect to your solar panel (<=6V max!), here is what we tested with:
https://www.adafruit.com/product/1390 (Micro USB shell)
http://www.exp-tech.de/seeed-studio-0-5w-solar-panel (smallest, use larger if needed)

more details on solar power and ESP8266 tuning on here:
https://www.hackster.io/fablabeu/esp8266-thing-by-sparkfun-982bc6

Tip: there are couple of traces on the PCB that you can cut to reduce over all power consumption (NeoPixel, USB/UART bridge), with that you can get down to 20uA sleep current (see also deep-sleep code)

c. Batterie only option – this is a hack just feeding power from 2xAAA batteries, no LiPo no USB power

Close this solder jumper (left)

Cut this trace (right)

Mount battery holder clips – *not soldering needed ! – yes and we us NiMH rechargeable AAA with a 1.2V level, they are working great (on your own risk as out of spec, but as said it is a nice hack!)*

Attention in this hack there is **NO** reverse polarity check,
make sure you insert the batteries right way!

d. adding GPS module for trace and trace applications:

It is simple just need the Adafruit ultimate GPS a listed in the parts section,
will add picture with orientation soon!

Hacks:

Some time you want to power a sensor with 5V – while the I/O is 3.3V ready. In this case you can jumper the USB 5V to the header – like follows:

(this will work of course only if the OCTOPUS is powered by 5V / USB)

More resources:

Btw there is a hackathon, which is going to use OCTOPUS with some more instructions on coding and Arduino(TM) IDE examples:

Your first example, using the BME280 and NeoPixel to detect humidity!

```
#define PIN 13 //NeoPixl

#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME280 bme; // I2C
Adafruit_NeoPixel strip = Adafruit_NeoPixel(1, PIN, NEO_GRB + NEO_


// Fill the dots one after the other with a color
void colorWipe(uint32_t c, uint8_t wait) {
  for(uint16_t i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, c);
    strip.show();
    delay(wait);
  }
}

void setup() {
  Serial.begin(9600);

  strip.begin();
  strip.show(); // Initialize all pixels to 'off'

  delay(1000);

  Serial.println(F("BME280 test"));
  if (!bme.begin()) {
    Serial.println("Could not find a valid BME280 sensor, check wi
    while (1);
  }

  delay(1000);

}

```

```
void loop() {

    Serial.print("Temperature = ");
    Serial.print(bme.readTemperature());
    Serial.println(" *C");

    Serial.print("Pressure = ");

    Serial.print(bme.readPressure() / 100.0F);
    Serial.println(" hPa");

    Serial.print("Approx. Altitude = ");
    Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
    Serial.println(" m");

    Serial.print("Humidity = ");
    Serial.print(bme.readHumidity());
    Serial.println(" %");

  if (bme.readHumidity() < 30) { colorWipe(strip.Color(255, 0, 0),
    colorWipe(strip.Color(0, 0, 255), 50); // Green}
  }

    Serial.println();
}
```

A more advanced sample using the Wifi connect and talking to the Blynk App!

```
#define BLYNK_PRINT Serial    // Comment this out to disable prir
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <Adafruit_NeoPixel.h>
#include <SimpleTimer.h>

#include <stdint.h>
#include "SparkFunBME280.h"
//Library allows either I2C or SPI, so include both.
#include "Wire.h"
#include "SPI.h"

//Global sensor object
BME280 mySensor;

SimpleTimer timer;

//////////////////////
// Blynk Settings //
//////////////////////
char BlynkAuth[] = "YOURBLYNKKEY";
char WiFiNetwork[] = "YOURSSID";
char WiFiPassword[] = "YOURPWD";

////////////////////////
// Hardware Settings //
////////////////////////
#define WS2812_PIN 13 // Pin connected to WS2812 LED
#define BUTTON_PIN 0
#define LED_PIN    0
Adafruit_NeoPixel rgb = Adafruit_NeoPixel(1, WS2812_PIN, NEO_GRB

BLYNK_WRITE(V0) // Handle RGB from the zeRGBa
{
```

```
  if (param.getLength() < 5)
    return;

  byte red = param[0].asInt();
  byte green = param[1].asInt();
  byte blue = param[2].asInt();

  uint32_t rgbColor = rgb.Color(red, green, blue);
  rgb.setPixelColor(0, rgbColor);
  rgb.show();
}

// This function sends Arduino's up time every second to Virtual
// In the app, Widget's reading frequency should be set to PUSH.
// that you define how often to send data to Blynk App.
void myTimerEvent()
{
  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V5, mySensor.readTempC());
}

void setup()
{
  // Initialize hardware
  Serial.begin(9600); // Serial
  rgb.begin(); // RGB LED
  pinMode(BUTTON_PIN, INPUT); // Button input
  pinMode(LED_PIN, OUTPUT); // LED output

  // Initialize Blynk
  Blynk.begin(BlynkAuth, WiFiNetwork, WiFiPassword);

  // Setup a function to be called every second
  timer.setInterval(1000L, myTimerEvent);

  //***Driver settings********************************//
  mySensor.settings.commInterface = I2C_MODE;
  mySensor.settings.I2CAddress = 0x77;
  mySensor.settings.runMode = 3; //Normal mode
  mySensor.settings.tStandby = 0;
  mySensor.settings.filter = 0;
  mySensor.settings.tempOverSample = 1;
  mySensor.settings.pressOverSample = 1;
  mySensor.settings.humidOverSample = 1;

  Serial.print("Program Started\n");
  Serial.print("Starting BME280... result of .begin(): 0x");

  //Calling .begin() causes the settings to be loaded
  delay(10);  //Make sure sensor had enough time to turn on. BME2
  Serial.println(mySensor.begin(), HEX);

  Serial.print("Displaying ID, reset and ctrl regs\n");

  Serial.print("ID(0xD0): 0x");
  Serial.println(mySensor.readRegister(BME280_CHIP_ID_REG), HEX);
  Serial.print("Reset register(0xE0): 0x");
  Serial.println(mySensor.readRegister(BME280_RST_REG), HEX);
  Serial.print("ctrl_meas(0xF4): 0x");
  Serial.println(mySensor.readRegister(BME280_CTRL_MEAS_REG), HEX
  Serial.print("ctrl_hum(0xF2): 0x");
  Serial.println(mySensor.readRegister(BME280_CTRL_HUMIDITY_REG),
}
```

```
99
100 void loop()
101 {
102   // Execute Blynk.run() as often as possible during the loop
103   Blynk.run();
104   timer.run(); // Initiates SimpleTimer
105 }
```

If you want to become even more advanced look into sleep mode, specially interesting when running on batteries or solar power. Also this one is important when you are running on LiPo and want to charge it fully, give headroom during sleep.

```
1  unsigned int deepSleepSeconds = 30;      // Number of seconds for
2  void setup() {
3  pinMode(0, OUTPUT);
4  }
5  void loop() {
6  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage
7  delay(1000);              // wait for a second
8  digitalWrite(13, LOW);    // turn the LED off by making the voltag
9  delay(1000);              // wait for a second
10 ESP.deepSleep(1000000 * deepSleepSeconds, WAKE_RF_DEFAULT); // GPI
11 }
```

Here is an example with the Adafruit 7-Segment FeatherWing, and NTP -> Building a clock!

```
1  #include <SPI.h>                                    // SPI interface AP
2  #include <Wire.h>                                   // Wire support lib
3  #include <ESP8266WiFi.h>
4  #include "Adafruit_LEDBackpack.h"                   // Support for the
5  #include "Adafruit_GFX.h"                           // Adafruit's graph
6  #include <WiFiUdp.h>
7
8  //#############################################################
9  // Globals
10 //#############################################################
11
12 char ssid[] = "YOURSSID";
13 char pass[] = "YOURPWD";
14
15 #define TIME_24_HOUR    true
16 #define DISPLAY_ADDRESS 0x70
17
18 // Create display object
19 Adafruit_7segment clockDisplay = Adafruit_7segment();
20
21 int hours = 0;                          // Track hours
22 int minutes = 0;                        // Track minutes
23 int seconds = 0;                        // Track seconds
24 int tzOffset = +1;                      // Time zone offset
25
26 bool blinkColon = false;                // Track the status of the cc
27
28 unsigned int localPort = 2390;          // Local port to listen for l
29 IPAddress timeServer(129,6,15,28);  // time.nist.gov NTP server
30 const int NTP_PACKET_SIZE = 48;     // NTP time stamp is in the f
31 byte packetBuffer[NTP_PACKET_SIZE]; // Buffer for incoming and ou
32
33 WiFiUDP EthernetUdp;
```

```
34
35  //##############################################################
36  // Functions
37  //##############################################################
38
39
40  // send an NTP request to the time server at the given address
41  unsigned long sendNTPpacket(IPAddress& address)
42  {
43    //Serial.println("1");
44    // set all bytes in the buffer to 0
45    memset(packetBuffer, 0, NTP_PACKET_SIZE);
46    // Initialize values needed to form NTP request
47    // (see URL above for details on the packets)
48    //Serial.println("2");
49    packetBuffer[0] = 0b11100011;   // LI, Version, Mode
50    packetBuffer[1] = 0;     // Stratum, or type of clock
51    packetBuffer[2] = 6;     // Polling Interval
52    packetBuffer[3] = 0xEC;  // Peer Clock Precision
53    // 8 bytes of zero for Root Delay & Root Dispersion
54    packetBuffer[12]  = 49;
55    packetBuffer[13]  = 0x4E;
56    packetBuffer[14]  = 49;
57    packetBuffer[15]  = 52;
58
59    //Serial.println("3");
60
61    // all NTP fields have been given values, now
62    // you can send a packet requesting a timestamp:
63    EthernetUdp.beginPacket(address, 123); //NTP requests are to pc
64    //Serial.println("4");
65    EthernetUdp.write(packetBuffer, NTP_PACKET_SIZE);
66    //Serial.println("5");
67    EthernetUdp.endPacket();
68    //Serial.println("6");
69  }
70
71  void setup() {
72
73    Serial.begin(115200);                              // Start the se
74    Serial.println("Clock starting!");                 // Start the cl
75
76    // Set up the display.
77    clockDisplay.begin(DISPLAY_ADDRESS);
78      clockDisplay.writeDisplay();
79
80    WiFi.mode(WIFI_STA);
81
82    // Attempt to conect to the WiFi network.
83    Serial.println("Connecting to WiFi netowrk.");
84    while (WiFi.status() != WL_CONNECTED) {
85      WiFi.begin(ssid, pass);                 // Connect to WPA2 netwc
86      uint8_t timeout = 10;                            // Set a timeou
87      while (timeout && (WiFi.status() != WL_CONNECTED)) {
88        timeout--;                                     // Decrement ti
89        delay(1000);                                   // Delay for or
90      }
91    }
92
93    Serial.println("Connected to network.");
94
95    EthernetUdp.begin(localPort);                              // Oper
96  }
97
```

```
//############################################################
// Loop - main
//############################################################

// This is one of the two standard functions for every Arduino pr
// other being setup(). This one runs continuously, forever, and
// Arduino program code.

void loop() {
  // Refresh the time at the top of every hour, or every five mir
  // the clock drift on the bare Feather M0 is pretty wicked.
  if ((minutes == 0) || ((minutes % 5) == 0)) {
    sendNTPpacket(timeServer); // send an NTP packet to a time se
   // wait to see if a reply is available
   delay(1000);
   if ( EthernetUdp.parsePacket() ) {
     Serial.println("packet received");
     // We've received a packet, read the data from it
    EthernetUdp.read(packetBuffer, NTP_PACKET_SIZE); // read the

     //the timestamp starts at byte 40 of the received packet anc
     // or two words, long. First, esxtract the two words:

     unsigned long highWord = word(packetBuffer[40], packetBuffer
     unsigned long lowWord = word(packetBuffer[42], packetBuffer[
     // combine the four bytes (two words) into a long integer
     // this is NTP time (seconds since Jan 1 1900):
     unsigned long secsSince1900 = highWord << 16 | lowWord;
     Serial.print("Seconds since Jan 1 1900 = " );
     Serial.println(secsSince1900);

          // now convert NTP time into everyday time:
     Serial.print("Unix time = ");
     // Unix time starts on Jan 1 1970. In seconds, that's 220898
     const unsigned long seventyYears = 2208988800UL;
     // subtract seventy years:
     unsigned long epoch = secsSince1900 - seventyYears;
     // print Unix time:
     Serial.println(epoch);

     // print the hour, minute and second:
     Serial.print("The UTC time is ");       // UTC is the time a
     hours = ((epoch  % 86400L) / 3600);     // print the hour (8
      hours += tzOffset;                     // Calculate the tim
      if (hours < 0) { hours = 24 + hours; } if (hours > 23) {
        hours = hours - 23;
      }
     Serial.print(hours); // print the hour
     Serial.print(':');
     minutes = ((epoch % 3600) / 60);
     if (minutes < 10 ) {
       // In the first 10 minutes of each hour, we'll want a leac
       Serial.print('0');
     }
     Serial.print(minutes);                  // print the minute (
     Serial.print(':');
     seconds = (epoch % 60);                // print the second
     if ( seconds < 10 ) { // In the first 10 seconds of each mir
      displayValue -= 1200;
    }
    else if (hours == 0) {
      displayValue += 1200;
    }
  }
```

```
162
163     // Print the time on the display
164     clockDisplay.print(displayValue, DEC);
165
166     // Add zero padding when in 24 hour mode and it's midnight.
167     // In this case the print function above won't have leading 0's
168     // which can look confusing.  Go in and explicitly add these ze
169     if (TIME_24_HOUR && hours == 0) {
170       // Pad hour 0.
171       clockDisplay.writeDigitNum(1, 0);
172       // Also pad when the 10's minute is 0 and should be padded.
173       if (minutes < 10) { clockDisplay.writeDigitNum(2, 0); } } //
174       seconds = 0;
175       minutes += 1;
176       // Again if the minutes go above 59 then the hour should incr
177       // the minutes should wrap back to 0.
178       if (minutes > 59) {
179         minutes = 0;
180         Serial.println("Minutes set to zero - should query NTP on r
181         hours += 1;
182         // Note that when the minutes are 0 (i.e. it's the top of o
183         // then the start of the loop will read the actual time fro
184         // again.  Just to be safe though we'll also increment the
185         // back to 0 if it goes above 23 (i.e. past midnight).
186         if (hours > 23) {
187           hours = 0;
188         }
189       }
190     }
191   }
```

If you want to pimp up the Octopus you can download a laster cut profile for a plexi housing!

Schematic for your reference. PCB can be found on oshpark.com (external)

39 Kommentare zu “OCTOPUS”

Philipp Jenke sagt:

Dezember 18, 2016 um 7:37 am Uhr

Hallo Octopus-Team,

kann ich den bestehenden „Neopixels-Strip aus zwei LEDs“ verlängern? Ansonsten ist der Neopixels-Anschluss, wenn ich es richtig sehe, wegen der 3,3V (Octopus/ESP8266) vs. 5V (Neopixels) nicht ganz trivial, oder?

Vielen Dank, Philipp

FabLab sagt:

Dezember 18, 2016 um 3:34 pm Uhr

… ganz einfach auf dem Grove (z.B. rechts) die 3.3V und GND anzapfen und an I/O (Kroko, Bananenbuchse) den IN der NeoPixel Kette anschliessen! funktioniert … der Hinweis: ist ausserhalb der Spezifikation der WS2812B und eigenes Risiko, verbraucht dabei (3.3V statt 5V) ziemlich wenig Strom 😉 trotzdem immer auch den Stromverbrauch achten, sonst ggfs extern 3.3V beisteuern, erstaunlich auch wie viele NeoPixel mit 2xAAA Batterien leuchten … Trade off bei 3.3V leuchtet die blaue LED naturgemäß nicht so kräftig!

Tomas Muehlhoff sagt:

Dezember 30, 2016 um 10:48 pm Uhr

Hallo,
ich hab die 5V direkt hinter R22 am USB Port abgegriffen…. klappt auch…

Tomas Muehlhoff sagt:

Dezember 30, 2016 um 10:48 pm Uhr

Sorry… Ihr arbeitet mit Akkus 😉

Philipp Jenke sagt:

Dezember 18, 2016 um 6:35 pm Uhr

Prima, danke!

Martin SIpek sagt:

Dezember 19, 2016 um 3:15 pm Uhr

Wie bekommt man es fertig bestückt??

FabLab sagt:

Dezember 23, 2016 um 5:58 pm Uhr

derzeit auf https://www.tindie.com/products/FabLab/octopus-the-iot-badge/

Jacek Jakubowski sagt:

Dezember 22, 2016 um 6:58 pm Uhr

Hallo,

ich habe mit Begeisterung im Make-Heft über den IoT.Octopus gelesen. Nun will ich für meinen Neffen (10J) ein Experimentierset zusamenstellen. Das Board habe ich bereits und ich kann es mit Arduino-IDE prgrammieren. Allerings würde ich meinem Neffen gerne die Scratch Programmierung mit Ardublocks anbieten. Finde aber niergendwo die ardublockIoT.jar. Ist jemandem hier die Quelle dafür bekannt? Ausserdem ist auch niergendwo ein Schaltplan aufzutreiben. Hätte das vielleicht jemand?

Danke schon mal

Jacek Jakubowski sagt:

Dezember 22, 2016 um 7:15 pm Uhr

Ups… der Schaltplan hat sich so eben erledigt. Steht am Anfang dieser Kommentar Reihe. Dann brauch ich aber immer noch ardublockIoT.jar.

FabLab sagt:

Dezember 23, 2016 um 5:59 pm Uhr

ja schaltplan ganz untern auf der Seite!

FabLab sagt:

Dezember 23, 2016 um 5:57 pm Uhr

hoffe der Link kommt noch rechtzeitig:
https://www.dropbox.com/s/iqyngbentzqeodu/IoT2016Final.zip?dl=0

Jacek Jakubowski sagt:

Dezember 23, 2016 um 9:43 pm Uhr

Vielen Dank. Das gibt morgen eine menge Freude

Elke sagt:

Januar 1, 2017 um 11:23 am Uhr

Schade, alles in englisch.
Ich bin auf der Suche für unsere Schule nach geeigneter Hardware für den leichten Einstieg in Programmierung, Elektronik, Digitalisierung. Lego Mindstorm, Arduino … oder könnte es der Octopus werden? Etwa ab Klasse 7 für interessierte Schüler.
Mir ist klar, dass in der Informatik die Hauptsprache Englisch ist, insbesondere bei der Suche nach Informationen im Internet. Aber gerne würde ich den Schülern Material zum selbständigen Einlesen geben. Einige durchaus interessierte Schüler wären dann leider sicherlich abgeschreckt, wenn komplizierte Sachverhalte sofort nur in der Sprache Englisch präsentiert werden.
Da in der neuen Make-Zeitschrift stand, in Deutschland entwickelt und produziert habe ich auch mehr deutschsprachige Unterstützung erhofft, gerade auch, weil im Artikel der Focus auf guter Ausbildung und Heranführung zum Thema und nicht der reine Kommerz stand. Der notwendige englischsprachige Hintergrund kommt dann von ganz alleine.
Frage: Wird es von den deutschsprachigen Entwicklern auch deutschsprachige Anleitungen geben?
Vielleicht habe ich sie auch nur noch nicht gefunden.
Mit herzlichen Grüßen und ein schönes Jahr 2017

FabLab sagt:

Januar 1, 2017 um 11:34 am Uhr

Hallo Elke,

absolut … wir haben alles lokal entwickelt und dokumentiert. Sind aber kompatibel zu den bekannten Communities um Adruino(TM) und Adafruit. Hier die Links zur Projektbeschreibung:

http://div-konferenz.de/events/hackathon/
Grafische „Programmierung“:
http://deutschland-intelligent-vernetzt.org/app/uploads/sites/4/2016/09/IoT_Hackathon_Ardublock_Version.pdf
und für Profis 😉
http://deutschland-intelligent-vernetzt.org/app/uploads/sites/4/2016/09/IoT_Hackathon_C_Code_Version.pdf

In den Dokumenten sind einige Beispiele zum Einstieg enthalten, aber auch anspruchsvollere Anwendungen als Grundlage für eigene Projekte.
Viele Spass und einen Gutes Neues 2017 mit vielen spannenden Projekten!
Guido

PS: bei Fragen gerne melden!

Jens Stolze sagt:

Januar 17, 2017 um 1:53 pm Uhr

Moin moin, ich hätte auch Interesse, den Octopus im Unterricht einzusetzen. Für meine SchülerInnen bräuchte ich aber die Ardublockumgebung. Leider funktioniert der obige Link zum Download nicht richtig, bei 314 von 551 MB ist Schluss ;-(Könnten wir vielleicht einen Link nur auf die Datei ardublockIoT.jar bekommen, die ist ja nur einige MB groß.

cu, Jens

—
IoT2016Final.zip
https://dl.dropboxusercontent.com/content_link/xxv4lMvwhzc7HlWIW0uosoyxDy7Z6PZ5sjePLzxF3reZaH9tQPtrtMCDeEVALug3/file?dl=1
0 Byte/s – 314 MB von 551 MB
—

FabLab sagt:

Januar 18, 2017 um 11:17 am Uhr

Hier ein aktualisierter Link für die Entwicklungsumgebung.
https://www.dropbox.com/s/8ea2943eyxat8m1/IoT2016Final.zip?dl=0

Das Ardublock – Jar ohne IDE (für Selbstinstallierer)
https://www.dropbox.com/s/5oxw4d3we3t3g8n/ardublock-IoT.jar?dl=0

Andreas sagt:

März 2, 2017 um 9:26 pm Uhr

Hallo,

Ich habe mir den octopus jetzt auch bestellt und bin schon ganz gespannt wenn er das erstmal läuft.
Ich würde gerne folgenden Sensor daran betreiben um die luftqualität in der Wohnung zu überwachen.
http://sandboxelectronics.com/?product=mh-z16-ndir-co2-sensor-with-i2cuart-5v3-3v-interface-for-arduinoraspeberry-pi
Der Sensor benötigt als versorgungsspannung aber 5V! Kann ich die auf dem Board irgendwo

abgreifen, oder brauche ich ein externes Netzteil?

Viele Grüße nach Stuttgart aus Fellbach
Andreas

FabLab sagt:

März 3, 2017 um 6:45 am Uhr

ja klar – auf der Rückseite sind zwei Jumper (Lötbrücken) die eine ist defaukt auf 3.3V gesetzt, kann aber aufgetrennt werden und mit einer Lötbrücke auf dem mit „5“ markiertem Pad auf 5V gesetzt werden – Achtung es dürfen niemals beide Brücken gesetzt (verbunden) werden! Bilder folgen

Andreas sagt:

März 3, 2017 um 2:28 pm Uhr

Super! Dann warte ich mal af die Bilder, nicht das noch etwas zerschossen wird
Vielen Dank vorab.

FabLab sagt:

März 28, 2017 um 10:04 am Uhr

Bilder sind endlich online 😉

Carsten sagt:

April 6, 2017 um 4:23 am Uhr

Hallo,

ich habe auf das Board die NodeMcu Firmware geflasht (siehe http://nodemcu.com/index_en.html bzw. https://nodemcu-build.com).
Funktioniert soweit gut, lässt sich auch prima mit dem ESPlorer Tool verwenden.
Nur bekomme ich den integrierten BME280 Sensor nicht angesprochen.

NodeMcu bietet für den Sensor ein extra Modul:
https://nodemcu.readthedocs.io/en/master/en/modules/bme280/

Die Initialisierungs-Methode „bme280.init()“ schlägt immer fehl (als Parameter SDA und SCL habe ich 3,4 bzw. 4,5 verwendet). Als Debug Log-Ausgabe kommt nur „No ACK on address 0x76“ und „No ACK on address 0x77“ (die 0x77 sollte es laut Board-Aufdruck ja eigentlich sein).
Hab auch schon andere, komplett Lua-basierte Module für den BME280 probiert. Ging auch nicht.
Bei einem Test mit der Arduino-IDE funktioniert der Sensor.

Hat jemand vielleicht eine Idee, wie es funktioniert?

Danke im voraus.

FabLab sagt:

April 6, 2017 um 7:04 am Uhr

vielleicht hilft dass: https://www.hackster.io/fablabeu/esp8266-sensor-f4c9b1

Achim Kern sagt:

Juni 4, 2017 um 4:50 pm Uhr

Hallo – 5 Boards wurden heute geordert. Ich werde versuchen diese in mein LCARS SmartHome System zu integrieren. Interessieren würde ich mich noch für die Feinstaubmessungen und die Datenübermittlung mit LoRa.

Lothar Schüller sagt:

Juli 26, 2017 um 10:21 pm Uhr

Hallo,

Freue mich schon sehr auf das bestellte board…
Kurze Frage: Die links zur Ardublock-Erweiterungen gehen ins Leere… Wäre es möglich das Ardublock-jar nochmal zur Verfügung zu stellen?

Grüße aus Münsing,
Lothar

Dominique sagt:

August 8, 2017 um 2:05 pm Uhr

Hallo alle
Ich habe 2 Fragen:

1. Ich scheitere an folgender Fehlermeldung:
warning: espcomm_sync failed
An error occurred while uploading the sketch
error: espcomm_open failed
error: espcomm_upload_mem failed
Kommt anscheinend öfters bei ESP8266 vor. Bin ein totaler newbie! Wie kann ich diesen Fehler auf der octopus-platine einfach beheben?

2. Kann mir jemand einen Link geben für die arduinoblock-jar datei des iot-hachathon 2016?

FabLab sagt:

August 8, 2017 um 5:56 pm Uhr

zu 1. schau mal hier zur Installation ESP8266 in Arduino IDE – ich nutze
„Generic 8266 Module“
Flash Mode „DIO“
Flash Frequency „40MHz“
CPU Frequency „80MHz“
Debug port „disabled“
Debug level „Keine“
Reset Method „nodemcu“
Upload Speed „921600“
Port -> entsprechend auswählen (meist ungleich COM1 !)
-> die Einstellung sollte funktionieren

zu 2. link update kommt, wird vermutlich morgen 😉

Dominique sagt:

August 8, 2017 um 9:38 pm Uhr

Vielen Dank für die schnelle Hilfe!
Ich habe alles so eingestellt wie du oben beschrieben hast. Funktioniert trotzdem nicht.
Octopus hat anfänglich auch funktioniert mit der IDE 1.8.3. Als ich später irrtümlicherweise mit einer älteren IDE (1.6.11) einen sketch hochladen wollte kam die Fehlermeldung. Egal mit welcher IDE ich jetzt hochlade, es kommt immer dieselbe Fehlermeldung (warning: espcomm_sync failed).

Grüsse aus Zürich

FabLab sagt:

August 10, 2017 um 4:58 pm Uhr

Hallo Dominique – IDE Version ist egal … mmmh, Ferndiagnose ist immer schwierig. Ggfs mal kurz vor dem upload (bei ca 90%) den REST Button auf dem Board betätigen, sollte der Autoreset nicht durchgehen (unwahrscheinlich) – Für ganz harte Fälle gibt es den https://github.com/nodemcu/nodemcu-flasher oder ähnlich (gibt es auch als py) – flash the firmware neu … Probier mal mit dem Resettaster!

FabLab sagt:

August 8, 2017 um 8:43 pm Uhr

zu 2. Ardublock for #IoT OCTOPUS: https://seafile.rlp.net/f/22450fb8d7924eceb3c8/?dl=1
(attention this is large >100MB)

Dominique sagt:

August 8, 2017 um 10:12 pm Uhr

Vielen Dank für das File. Habe die ardublock-ioT.jar-Datei am richtigen Ort platziert – leider ohne Erfolg. Ich kann ardublock nach Aufstart der IDE im tools-menu nicht finden. Wenn ich aber andere ardoblock-jars platziere, kann ich ardublock im tools-menu sehen und auswählen.
Langsam beginne ich an mir zu zweifeln….
Was mache ich falsch?
Grüsse aus Zürich

FabLab sagt:

August 10, 2017 um 4:54 pm Uhr

Hallo Doninique – hier das Jar als Zip – magst Du das ausprobieren? http://fab-lab.eu/wp-content/uploads/2017/08/ardublock-IoT.jar_.zip
sonst kann es auch sein, dass der Pfad zu lang wird – ggfs mal das Arduino Verzeichnis direkt auf C: legen … aber probier erst mal das aktuelle jar.
Daumen drück!

Thomas Daniels sagt:

Februar 11, 2018 um 1:55 pm Uhr

Hallo,
leider scheitere ich genau wie der Kollege Dominik weiter oben an derselben Fehlermeldung. Auch nach manueller Installation kann das board nicht gefunden werden und einen Port kann man auch nicht einstellen (Feld ausgegraut.
Ich arbeite hier unter Win 10 und parallels Desktop auf einem imac und ebenso nur unter win 10 auf einem Laptop.
Ich wollte eigentlich einen Satz octopusse für meinen Unterricht anschaffen, aber bekomme das board leider selbst nicht ans Laufen. In der Schule haben wir win7, ob es da funktioniert?
Freue mich über jeden Hinweis.

FabLab sagt:

März 26, 2018 um 3:09 pm Uhr

Einstellungen – wie beschrieben? Port gefunden? parallel Desktop muss den COM Port / USB weiterreichen,
sonst kann er nicht gefunden werden … der serial Port vom ESP wird mit einem Converter (CP2104x) auf
einen USB Port gemapped … dieser muss dann unter MAC gefunden werden (hierzu ggfs den CP2104 Treiber
und Mac installieren) und dann in parallel für Win10 frei geben?! … Unter reinem Windows kann man die
gepackte Arudruino Distribution mit Ardublock einfach kopieren … ggfs auch hier den CP2104 Treiber
sonst sollte es laufen 😉

FabLab sagt:

März 26, 2018 um 8:25 pm Uhr

hoffe dies hilft weiter: http://fab-lab.eu/octopus/octopus-toolchain-on-macos/

@stefferber sagt:

März 26, 2018 um 9:28 am Uhr

Hallo,

auf meinem Mac Mini OS X El Capitan 10.11.6 und Arduino 1.8.5 sieht das alles etwas anders aus. Ich schaffe es nicht, das Octopus board über USB zu flashen:
„Archiving built core (caching) in:
/var/folders/8r/hmm24sts0cg3h5tt8lvf1_1c0000gq/T/arduino_cache_80158/core/core_esp8266_esp8266_generic_CpuFrequency_80,ResetMethod_nodemcu,CrystalFreq_26,FlashFreq_40,FlashMode_dio,FlashSize_512K0,led_2,LwIPVariant_v2mss536,Debug_Disabled,DebugLevel_None____,FlashErase_none,UploadSpeed_115200_3a93be2442f0c79bb77e921b78da049b.a
Der Sketch verwendet 246271 Bytes (49%) des Programmspeicherplatzes. Das Maximum sind 499696 Bytes.
Globale Variablen verwenden 32244 Bytes (39%) des dynamischen Speichers, 49676 Bytes für lokale Variablen verbleiben. Das Maximum sind 81920 Bytes.
warning: espcomm_sync failed
error: espcomm_open failed
error: espcomm_upload_mem failed
error: espcomm_upload_mem failed“

gibt es irgendwo ein How-To für Macs?
Die Installation des USB „SiLabs CP2104 Drivers“ hat gut funktioniert. Daran liegt es wohl nicht.

Das Menü „Werkzeuge->Ardublock“ gibt es bei mir auch nicht.

FabLab sagt:

März 26, 2018 um 3:05 pm Uhr

Ardublock „Plug-In“ in die Arduino IDE, für Mac haben wir das noch nicht gebündelt. In der
Distribution verwenden wir die „portable“ Version d.h. Arduino bringt alle Libs und Ardublock
mit (für Windows) – auf Mac müsste man dass (noch) von Hand zusammenführen ;-(Zum CP2104
Driver … unter Arduino Tools den entsprechenden Port ausgewählt?

@stefferber sagt:

März 26, 2018 um 4:26 pm Uhr

Hallo FabLab, danke für die schnelle Antwort.

Ardublock „Plug-in“ geht dann erstmal nicht. Das ist für mich auch nicht so kritisch. Kann ja programmieren 😉

Der CP2104 Treiber ist installiert. Die Arduino IDE erkennt auch, ob der Octopus am USB angeschlossen ist. Ich vermute es liegt eher an den vielen Parameter [Werkzeuge -> Board: „Generic ESP8266 Module“] und darunter. Habt Ihr einen Screenshot von den Parametern, die funktionieren?

Gruß, @stefferber

FabLab sagt:

März 26, 2018 um 8:25 pm Uhr

here is the doc – http://fab-lab.eu/octopus/octopus-toolchain-on-macos/

FabLab sagt:

März 26, 2018 um 8:26 pm Uhr

wenn OS <=10.10 muss der legacy Treiber installiert werden ...