Playing With A $7 AI-Thinker ESP32-CAM IoT Development Board

Henry Dietz

Keeping Current talk, 16:30, March 17, 2021

University of Kentucky
Electrical & Computer Engineering
Abstract

**Abstract:** The Arduino IDE and C++ libraries make microcontrollers easy to use — and some compatible systems are now surprisingly small, cheap, and powerful. The $7 AI-Thinker ESP32-CAM development board is designed to allow Internet-of-Things (IoT) devices to use face recognition and other computationally demanding algorithms. In 27×40mm, this board provides a 240MHz dual-core 32-bit processor, 2MP camera, 802.11b/g/n WiFi and BlueTooth, as well as wired I/O interfaces, a microSD slot and low-power modes. I will present an overview of the ESP32-CAM, discuss how to work around some of its quirks, and briefly show a few sample applications. For example, a single ESP32-CAM was used to implement Lafodis160, the LArge FOmat DIgital Scanning camera I presented at Electronic Imaging 2021.
The ESP32-CAM As An IoT Camera

Not bad for a $7 programmable camera with WiFi!
The ESP32-CAM As An IoT Camera

- **How fast is it?** **Not video speeds**... but not terrible:
  - 1600x1200 UXGA ~6.2 FPS (frames per second)
  - 800x600 SVGA ~12.7 FPS
  - 400x296 CIF ~25.6 FPS

- **Rolling shutter at ~1/50s**
  - Typical for webcam, slow for a hand-held camera (1/fs rule)
  - Tiny lens can effectively magnify camera shake
  - Can show distortions within a frame
KISS-E: Kentucky’s Interchangeable-lens Small Sensor E-mount camera

- Interchangeable-lens camera with Sony E-mount
  - Use stand alone, capture to TF card using OLED live view
  - Use tethered via USB (not UVC protocol... yet)
  - Use as IoT webcam via 802.11 or Bluetooth

- 1600x1200 native resolution, RGB color, no integrated NIR filter
- 9.8X crop factor: 50mm lens gives view of 491mm on FF
KISS-E Build <$25 (without lens)

- $7 ESP32-CAM
- $3 SSD1306 OLED, 128x64
- $0.50 Switch, 12mm momentary SPST push button
- Power supply; either
  - $2.50 FT232RL with USB
  - $2 CR123/16340 battery + $3 5V boost converter
Lafodis160: LArge FOrmat Dlgmental Scanning, 160mm coverage circle

http://aggregate.org/DIT/Lafodis160
Sample B&W Capture

- One exposure, 1600x1200
- OV2640 JPEG
- Shallow DoF from 4x5 lens
Lafodis160 Build <$50 (without lens)

- **Scan resolution:** default 500MP @ 4x5 inch; 2.6GP max
- **Dynamic range:** 8-10EV; HDR to 20EV
- **Color:** RGB CFA, no integrated NIR filter
- **Scan speed:** currently <1MP/s; theoretical peak ~10MP/s
- **Electronics:** ESP32-CAM, two 28BYJ-48 with ULN2003
- **Capture control:** wireless via Bluetooth (it’s an IoT device!)
- **Firmware update:** wireless via 802.11 WiFi
- **Power:** 5V via USB connector from external source
- **Build equip.:** 180mm dia. x120mm tall 3D printer, wire wrap
Arduino IDE

- Arduino started in 2005 at Interaction Design Institute Ivrea, Italy (aka, IDII)
- Open-source HW/SW
- Cross development IDE
  - C/C++ “sketches”
  - setup()
  - loop()
Embedded Code Is Magic!

- Embedded systems are full of magic details...
  - What pins?
  - Hardware registers
  - Real-time issues

- Difficult to get started

- Really difficult to debug embedded systems
Embedded Code Is Magic!

- Embedded systems are full of magic details...
  - What pins?
  - Hardware registers
  - Real-time issues

- Open source Libraries!
- Open source Examples!
- Some Debug support
An Example: NTP-Set Clock

- Everything is open source
  - Example code
  - Libraries used

- Lots of magic hidden using C++ classes & overloading...
  e.g., `Serial.println()`

- IDE has some ICE-like features
  - USB programming / debug
  - OTA programming...
  Over The Air via 802.11
Arduinos Are Wimpy...?

- **ESP32-CAM** isn’t wimpy... nor is it an Arduino
- **Need USB-TTL adapter**, e.g., CP2102 or FT232RL (Future Technology Devices Inc. chip – FTDI)
- **Arduino IDE** needs add-on for ESP32:
  
  
AI-Thinker ESP32-CAM

- Compute (180-310mA, 5μA deep sleep)
  - 240MHz ESPRESSIF 32-bit dual-core LX6
  - Internal 520KB SRAM, external 4MB PSRAM
  - 4MB flash memory; up to 4GB (32GB?) TF card
  - Crypto HW for RNG, ECC, RSA, SHA-2, AES

- I/O facilities (including antenna)
  - 802.11 b/g/n/e/i
  - Bluetooth v4.2 BR/EDR and BLE
  - SPI, I2C, MMC, ADC, PWM...

- Camera (and white LED light on board)
  - Omnivision OV2640, 1600x1200 native RGB
  - 2.2μm square pixels, 10-bit ADC, HW JPEGs
  - Removable lens
AI-Thinker ESP32-CAM I/O Pins

- Pins are **wildly overloaded** with different functions

https://github.com/raphaelbs/esp32-cam-ai-thinker/blob/master/docs/esp32cam-pin-notes.md
How I Got 4+4 Output Pins

- **Overloading:** I use Pin 4 as Angle ULN2003 IN3... but it also controls the white LED and TF card
- **Desperation:** removed red LED and used that for my Angle ULN2003 IN2 signal
More Annoyances

• Can run off 3.3V, but runs better off 5V – especially when reprogramming the part

• Brownout detector is a little too aggressive:
  
  ```
  #include "soc/soc.h"
  #include "soc/rtc_cntl_reg.h"
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
  ```

• Some pins need to be in certain states to boot...

• OTA requires space; “ESP32 Dev Module” with:
  
  1.9MB APP with OTA, 190K SPIFFS

• PSRAM is too slow for large OV2640 images; use JPEG and `img_converters.h` for RGB888
Still, It’s Easy To Use

- The key is to **fully leverage source code for libraries and examples**…

- For Lafodis160, I needed to drive 2 stepper motors (that’s what those 8 output pins were needed for)
  - There are **several libraries, with examples, for driving steppers**... all **open source**
  - None of those was usable because they **always left power on**... including the white LED!
  - Looking at the source, it was easy to understand how the steppers had to be controlled and thus I **wrote my own library that is more efficient and implements power management**
New Stepper Library

// library interface description
class FourStep {
public:
    // constructors
    FourStep(int motor_pin_1, int motor_pin_2, int motor_pin_3, int motor_pin_4);

    // actions
    void Feedrate(long feedrate);  // set feedrate, steps/s
    void Move(long to);            // set togo
    long ToGo();                   // read togo, how many steps left to go?
    void Off();                    // immediately power down motor
    int TryStep();                 // try to step (powers on if needed)

private:
    unsigned long msperstep;      // delay between steps, in ms, based on speed
    unsigned long last;           // last time a step was taken
    long togo;                     // steps to go
    int ss;                        // state of stepper: 0, 1, 2, or 3
    int power;                     // is the stepper power on?

    // motor step power patterns
    const int pattern_1[4] = { HIGH, LOW, LOW, HIGH };
    const int pattern_2[4] = { LOW, HIGH, HIGH, LOW };
    const int pattern_3[4] = { HIGH, HIGH, LOW, LOW };
    const int pattern_4[4] = { LOW, LOW, HIGH, HIGH };

    // motor pin numbers:
    int motor_pin_1;
    int motor_pin_2;
    int motor_pin_3;
    int motor_pin_4;
};
3D Printing & Wiring Tricks

- ESP32-CAM usually comes with pins installed in the board
- Can remove pins & solder
- Can use wire wrap: a good overview is at
  - 3D-print cavity for board
  - 3D-print traceless PCB as part of 3D design
- Don’t know how to wire-wrap?

Conclusion

- **AI Thinker ESP32-CAM** is remarkably versatile
  - More flexible than Canon PowerShots using CHDK
  - Ignoring the camera, a powerful embedded controller

- I’m posting helpful hints at:
  
  [http://aggregate.org/DIT/ESP32CAM](http://aggregate.org/DIT/ESP32CAM)

- **Even if you hate IoT, HW built to support it is cool**
  - Small, cheap, surprisingly powerful, and versatile
  - You might not even need to solder (**wire wrap!**)
  - **Arduino community** provides the necessary magic