An ultra-low-cost large-format wireless IoT camera

Henry Dietz & Paul Eberhart

ISS-070 Poster, 13:00, January 19, 2021

University of Kentucky
Electrical & Computer Engineering

University of Kentucky

EI2021 ISS-070

An Ultra-Low-Cost* Large-Format Wireless IoT Camera

University of Kentucky

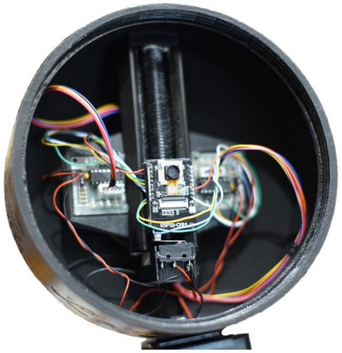
Aggregate.Org UNBRIDLED COMPUTING

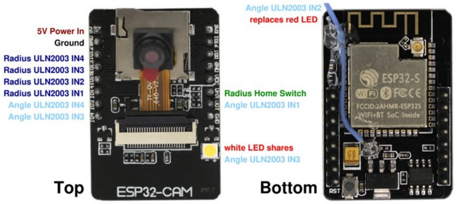
Henry Dietz & Paul Eberhart

Abstract

This paper documents the design, construction, and experimental evaluation of an ultra-low-cost large-format digital camera. Used lenses that cover formats up to 4x5 can be surprisingly inexpensive, but large-format image sensors are not. By combining 3D printing with cheap components developed for use in IoT (Internet of Things) devices, especially the sub-$10 ESP32-CAM, a digital scanning 4x5 camera capable of up to 2GP resolution can be constructed at very low cost. The camera created actually is a wireless IoT device, fully remote controllable via Bluetooth and WiFi. This camera also serves as a testbed for novel ways to improve capture quality for scenes that are not completely static during the scan interval, and a variety of methods employing unusual scan orderings and sensor region of interest (ROIs) manipulation are evaluated using it.

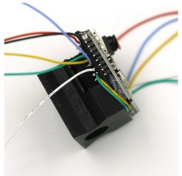
* under $100





Key Ideas

- **Sub-$10 AI-Thinker ESP32-CAM** IoT boardlet
  - **Omnivision OV2640 2MP camera**, lens removed; 2.2µm square pixels, 10-bit ADC
  - Arudino-compatible 32-bit dual core, 4.5MB SRAM; configured for 1.9MB APP with OTA, 190K SPIFFS
  - BlueTooth and 802.11 b/g/n WiFi
  - Implements camera & motion control logic… **tricky due to function-sharing of I/O pins**
- 3D printing tricks
  - Use wire wrap connections for ESP32-CAM, with **traceless PCB printed as part of design**
  - Linear motion rail
  - Use of printed screw threads for lens mount, focus



- Uses Angle, Radius rather than X, Y scan
  - Minimizes 3D-printed body part sizes
  - Allows full multi-aspect capture of lens coverage

Lafodis160: LArge FOrmat DIgital Scanning, 160mm coverage circle

- **Scan resolution:** default **500MP @ 4x5 inch**; theoretical peak **2.6GP**
- **Dynamic range:** 8-10EV; theoretical HDR limit is 20EV
- **Color:** RGB CFA, no integrated NIR filter
- **Scan speed:** currently <1MP/s; theoretical peak ~10MP/s
- **Construction:** 3D-printed body, linear rail, drive screw, electronics mounts, lens extension, lens focus thread, lens mount plate
- **Dimensions:** ~171mm diameter, ~190mm deep with 135mm lens
- **Weight:** 877g including $25 Wollensak 135mm f/4.5 enlarging raptar
- **Electronics: ESP32-CAM** and two **28BYJ-48 steppers** with **ULN2003** drivers
- **Capture control:** wireless via BlueTooth, host C++ program using OpenCV
- **Scan ordering:** by host, **dynamic angle/radius walk**
- **Firmware update:** wireless via WiFi, Arduino OTA compatible
- **Power:** 5V via USB connector from external source
- **Build materials cost:** approximately **$50** without lens
- **Build equipment/skills needed:** 3D printer with at least 180mm diameter by 120mm tall build volume, some wire-wrap & soldering required

Scanning & Stitching

- Created smarter low-level stepper library for Lafodis
  - Absolute position tracking, feedrate-controlled incremental stepping, & power management
  - Allows motor hold power off, which Lafodis needs to turn off white LED, ULN2003 LEDs
- Stitching based on ***"Senscape: modeling and presentation of uncertainty in fused sensor data live image streams"*** from EI2020, `https://doi.org/10.2352/ISSN.2470-1173.2020.14.COIMG-392`
  - **Integrated stitch & scan control is still an active research effort; Lafodis160 is a testbed**
  - Aligns each capture with master image (shift & rotate only – no lens corrections!)
  - **Each pixel value has a confidence associated with it**, confidences merged as well as values
    - Capture pixel confidence higher near sensor center and with better alignment
    - Disparate values (e.g., from scene motion during scan) reduce master pixel confidence
  - Basic scan order (e.g., **raster**, **spiral**, or **Hilbert curve**) can be altered by area confidence; ROI (Region Of Interest) and resolution can be adjusted for quick samples
  - Directed by host C++ OpenCV program using command language for Lafodis (via BlueTooth)

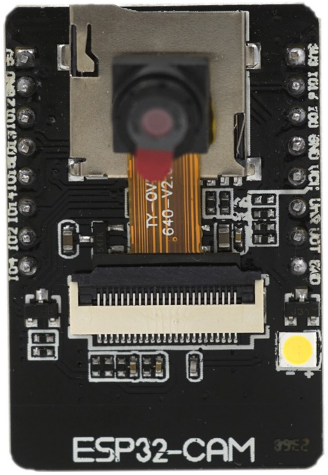
Sample B&W single OV2640 capture

Abstract

This paper documents the design, construction, and experimental evaluation of an ultra-low-cost large-format digital camera. Used lenses that cover formats up to 4x5 can be surprisingly inexpensive, but large-format image sensors are not. By combining 3D printing with cheap components developed for use in IoT (Internet of Things) devices, especially the sub-$10 ESP32-CAM, a digital scanning 4x5 camera capable of up to 2GP resolution can be constructed at very low cost. The camera created actually is a wireless IoT device, fully remote controllable via Bluetooth and WiFi. This camera also serves as a testbed for novel ways to improve capture quality for scenes that are not completely static during the scan interval, and a variety of methods employing unusual scan orderings and sensor region of interest (ROIs) manipulation are evaluated using it.

University of Kentucky

Sub-$10 AI-Thinker ESP32-CAM IoT boardlet



- **Omnivision OV2640 2MP camera**, lens removed; 2.2μm square pixels, 10-bit ADC
- Arudino-compatible 32-bit dual core, 4.5MB SRAM; configured for 1.9MB APP with OTA, 190K SPIFFS
- BlueTooth and 802.11 b/g/n WiFi
- Implements camera & motion control logic… tricky due to function-sharing of I/O pins

University of Kentucky

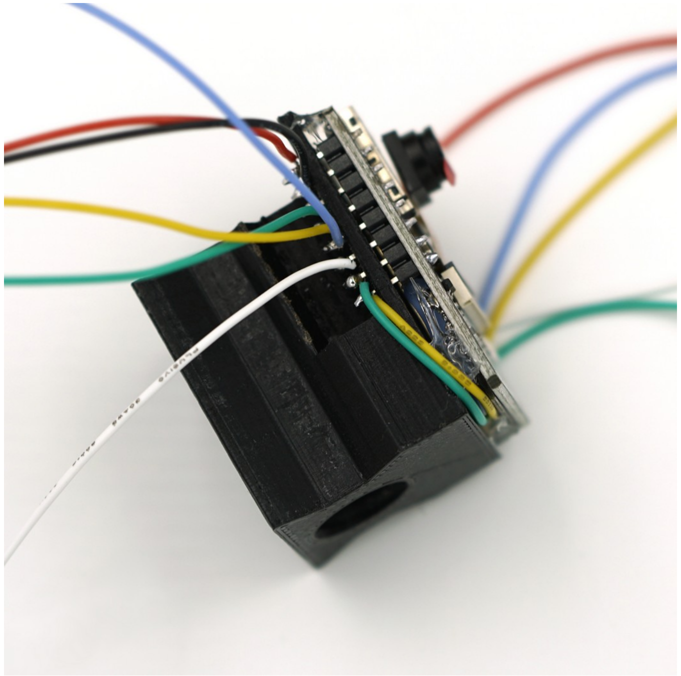
Sample B&W Capture

- One exposure, 1600x1200
- OV2640 JPEG (can shoot raw)
- Shallow DoF from 4x5 lens

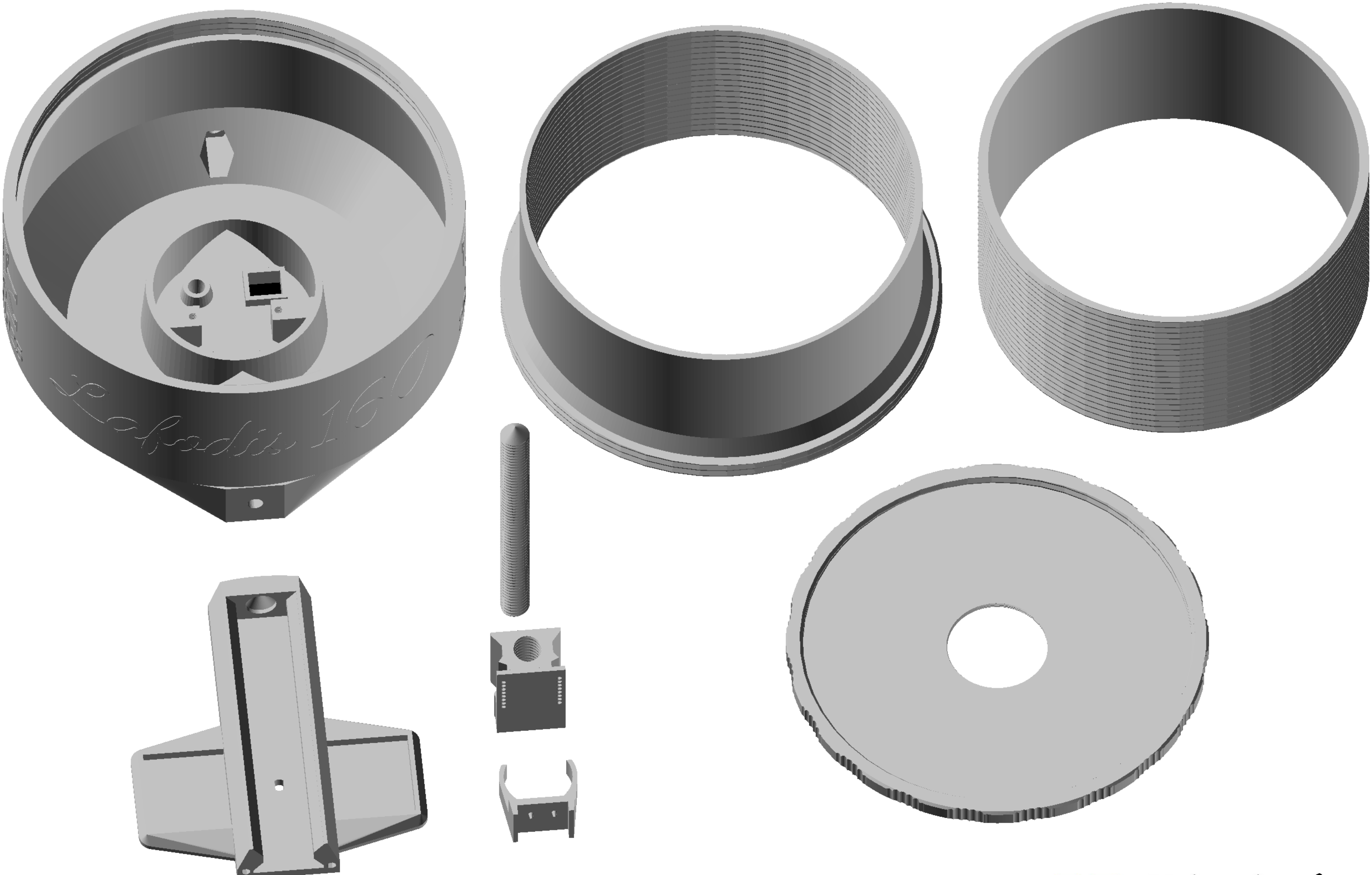
University of Kentucky

3D printing tricks

- Use wire wrap connections for ESP32-CAM, with **traceless PCB printed as part of design**
- Linear motion rail
- Use of printed screw threads for lens mount, focus

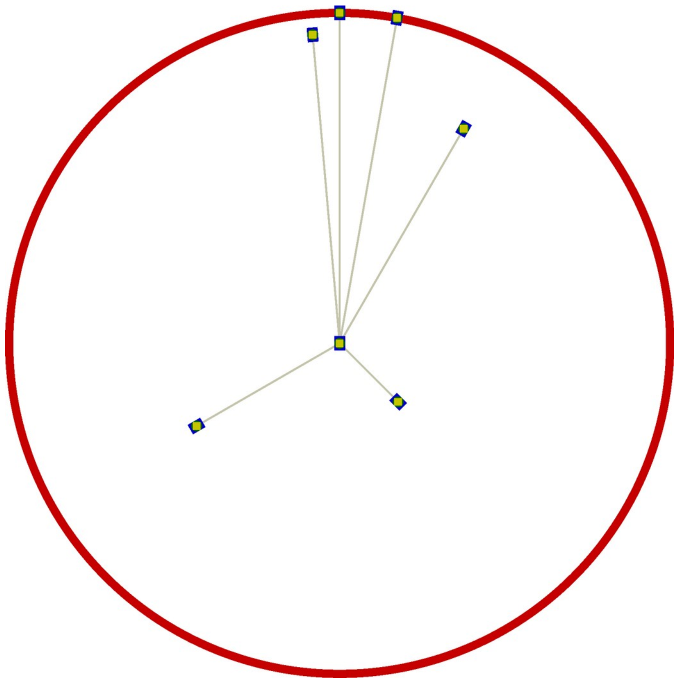


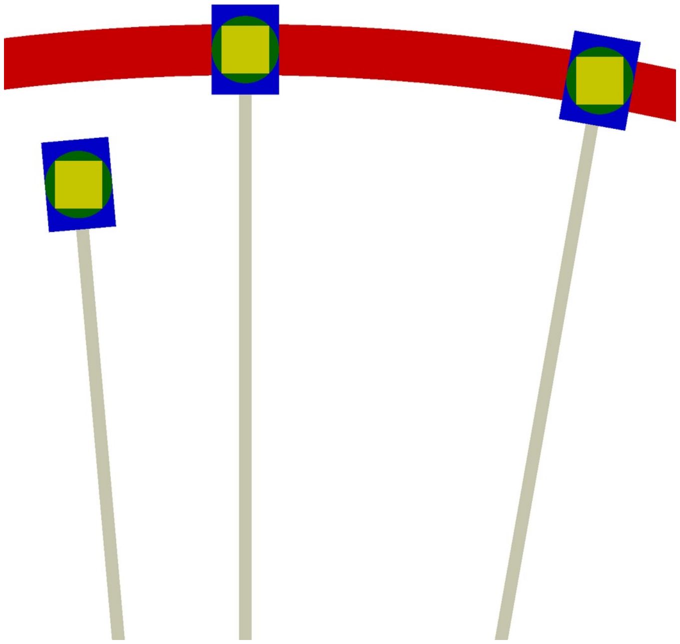
University of Kentucky



University of Kentucky

Uses Angle, Radius rather than X, Y scan





- Minimizes 3D-printed body part size
- Allows full multi-aspect capture of lens coverage

University of Kentucky

Lafodis160: LArge FOrmat DIgital Scanning,160mm coverage circle

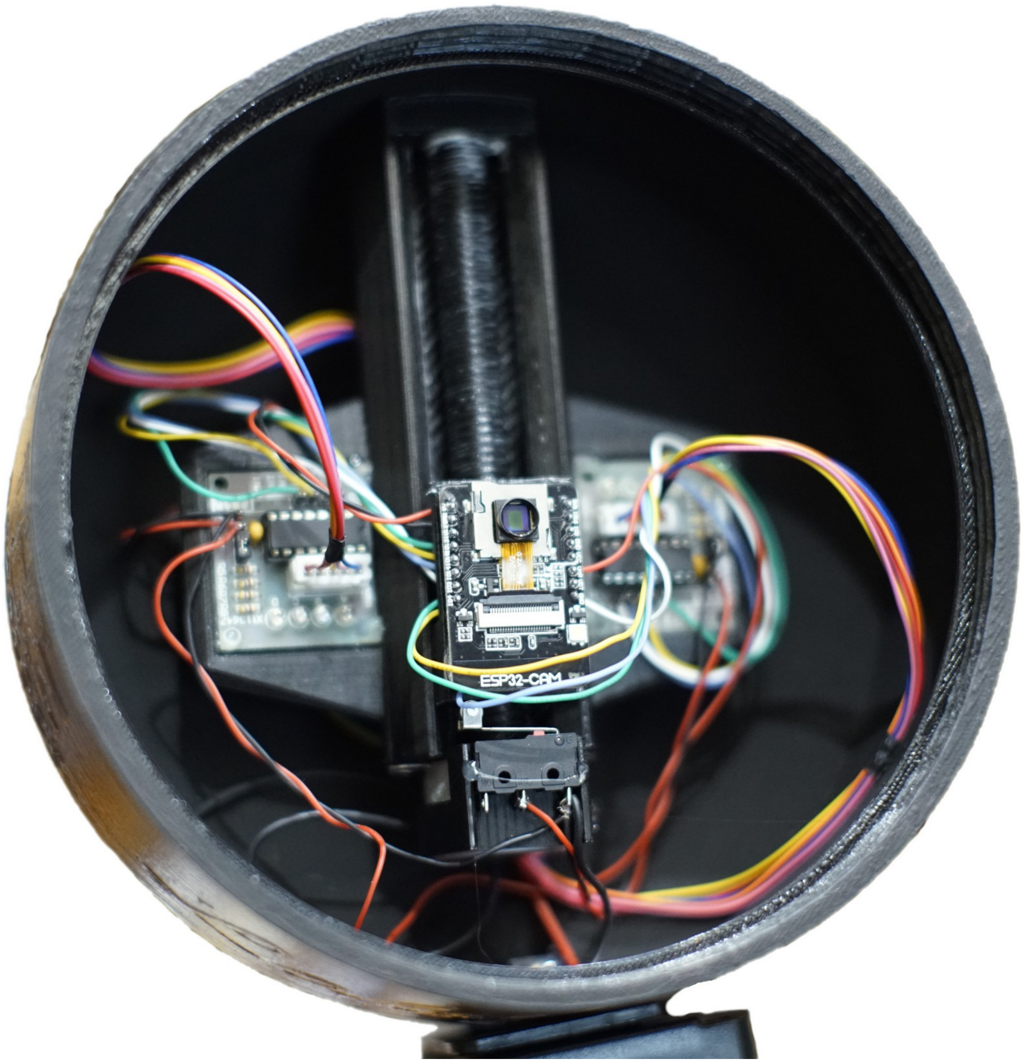
University of Kentucky

Lafodis160: LArge FOrmat DIgital Scanning,160mm coverage circle

- **Scan resolution:** default **500MP @ 4x5 inch**; theoretical peak 2.6GP
- **Dynamic range:** 8-10EV; theoretical HDR limit is 20EV
- **Color:** **RGB** CFA, no integrated NIR filter
- **Scan speed:** **currently <1MP/s**; theoretical peak ~10MP/s
- **Construction:** 3D-printed body, linear rail, drive screw, electronics mounts, lens extension, lens focus thread, lens mount plate
- **Dimensions:** ~171mm diameter, ~190mm deep (with 135mm lens)
- **Weight:** 877g (including $25 Wollensak 135mm f/4.5 enlarging raptar)

University of Kentucky

Lafodis160: LArge FOrmat DIgital Scanning,160mm coverage circle

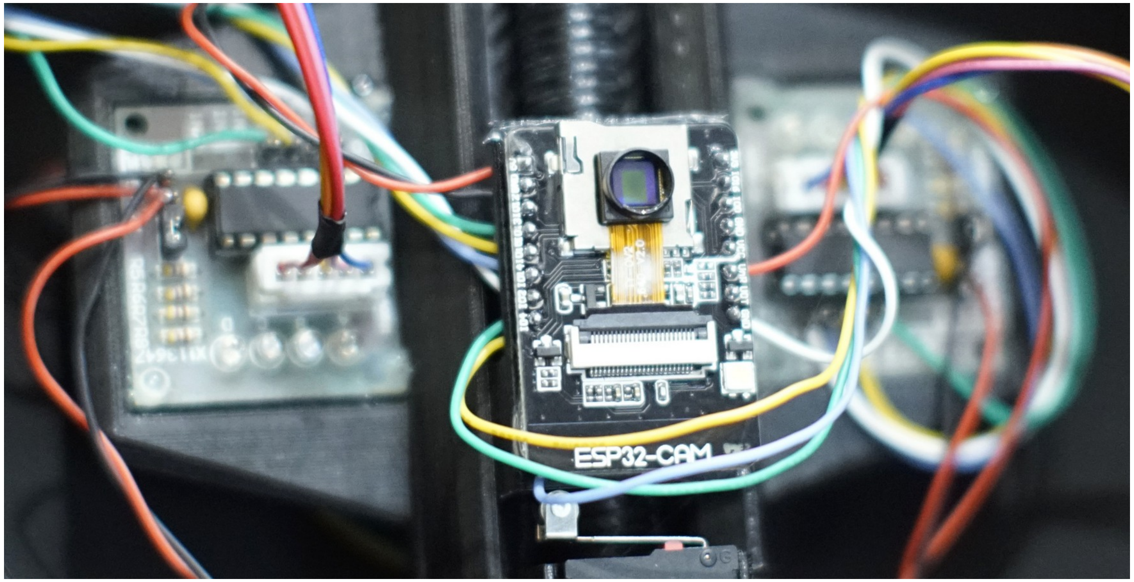


University of Kentucky

Lafodis160: LArge FOrmat DIgital Scanning,160mm coverage circle

- **Electronics:** **ESP32-CAM** and two **28BYJ-48 steppers with ULN2003 drivers**
- **Capture control:** wireless via BlueTooth,
  host C++ program using OpenCV
- **Scan ordering:** by host, **dynamic angle/radius walk**
- **Firmware update:** wireless via WiFi, Arduino OTA compatible
- **Power:** 5V via USB connector from external source
- **Build materials cost:** **approximately $50 without lens**
- **Build equipment/skills needed:** 3D printer with
  at least 180mm diameter by 120mm tall build volume,
  some wire-wrap & soldering required

Scanning



- Created smarter low-level stepper library for Lafodis
  - Absolute position tracking, feedrate-controlled incremental stepping, & power management
  - Allows motor hold power off, which Lafodis needs to turn off white LED, ULN2003 LEDs

University of Kentucky

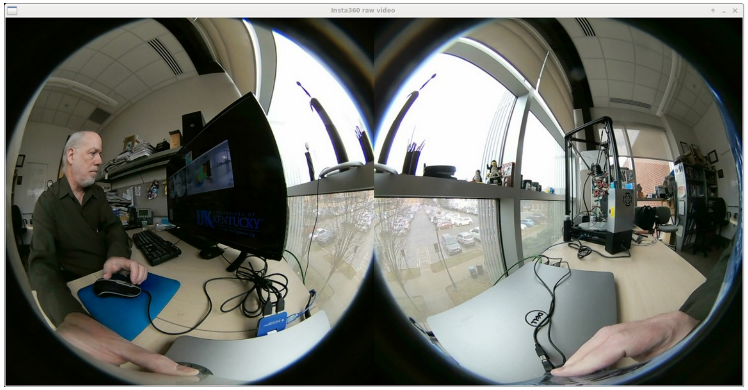
New Stepper Library

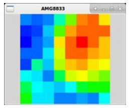
```
// library interface description
class FourStep {
  public:
    // constructors
    FourStep(int motor_pin_1, int motor_pin_2, int motor_pin_3, int motor_pin_4);

    // 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;
};
```

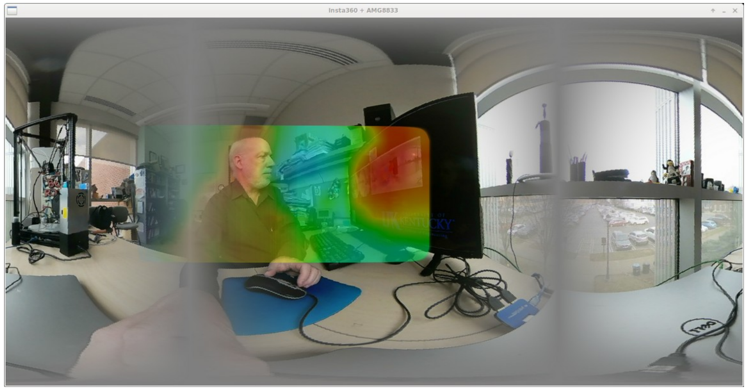
University of Kentucky

Stitching based on Senscape

https://doi.org/10.2352/ISSN.2470-1173.2020.14.COIMG-392







- **Each pixel value has a confidence associated with it**, confidences merged as well as values
  - Capture confidence higher near center, good alignment
  - Disparate values reduce master pixel confidence

University of Kentucky

Stitching & Scanning

- **Lafodis160 is a testbed** for integrated stitch & scan control
- Aligns each capture with master image
  (shift & rotate only – no lens corrections!)
- Basic scan order (e.g., raster, spiral, or Hilbert curve) can be altered by area confidence; ROI and resolution varied
- Directed by host C++ OpenCV program using command language for Lafodis (via BlueTooth)

Lafodis160 Remote Commands

```
#define L_ACK ';' // ack ending a command's output
#define L_NACK '/' // nack ending a command's output
#define L_CHK ',' // request ack when here

#define L_VERSION '?' // show version, YYYYMMDD
#define L_SPEED '$' // set step speed in RPM (feedrate)
#define L_SETHOME ':' // set here as home
#define L_GOHOME '.' // go home (based on switch)
#define L_WHERE '%' // where are we? (radius * 2048) + (angle % 2048)
#define L_GO '&' // go to (radius * 2048) + (angle % 2048)
#define L_GOA '<' // absolute goto angular position
#define L_INCA '>' // incremental goto angular position
#define L_GOR '!' // absolute goto radial position
#define L_INCR '^' // incremental goto radial position
#define L_IMAGE '*' // capture image, hex ends with ';'


#define L_READ '=' // read value of parameter, e.g., =P

#define L_MACRODEF '{' // start defining macro, e.g., {...}
#define L_MACROEND '}' // end macro def
#define L_MACRO '@' // apply macro, e.g., @

#define L_IGNORE '#' // ignore until end of line, comment
#define L_REMBEGIN '[' // begin remark (comment)
#define L_REMEND ']' // end remark (comment)
```

```
#define L_AELEVEL 'A' // set AE level, -2:2, 0 default
#define L_BRIGHTNESS 'B' // set brightness, -2:2, -2 default
#define L_CONTRAST 'C' // set contrast, -2:2, 2 default
#define L_DELAY 'D' // set ms delay for image to settle, 0:8000, 250 default
#define L_AGC 'E' // set AGC, 0:1, 0 default
#define L_EFFECT 'F' // set effect, 0:6, 0 (none) default
#define L_GAIN 'G' // set gain, 0:30, 5 (6X) default
#define L_HOLD 'H' // hold with power on motors? 0:1, 0 default
#define L_WBPC 'I' // set WPC & BPC, (0,2)+(0,1), 0 default
#define L_GAMMA 'J' // set raw gamma, 0:1, 1 default
#define L_AEC 'K' // set AEC sensor & DSP, (0,2)+(0,1), 0 default
#define L_LENS 'L' // set lens correction, 0:1, 1 default
#define L_AWBMODE 'M' // set AWB mode, 0:4, 0 (auto) default
#define L_NUMBER 'N' // set number of frames to sample before returning one, 0 default
#define L_ORIENT 'O' // set horizontal mirror & vertical flip, (0,2)+(0,1), 0 default
#define L_PIXEL 'P' // set pixel format, 0:7, 3 (PIXFORMAT_JPEG) default
#define L_QUALITY 'Q' // set quality, 10:63, 10 (best) default
#define L_RESOLUTION 'R' // set resolution, 0,3:10, 7 (800x600) default
#define L_SATURATION 'S' // set saturation, -2:2, 0 default
#define L_EXPOSURE 'T' // set exposure time, 0:1200, 204 default
#define L_DCW 'U' // DCW (Downsize EN), 0 default
#define L_VERBOSE 'V' // verbose messages?, 0 (no) default
#define L_AWB 'W' // set AWB & AWB Gain, (0,2)+(0,1), 0 default
#define L_FEEDR 'X' // set feedrate for radial position
#define L_FEEDA 'Y' // set feedrate for angular position
#define L_WHERE 'Z' // go to (radius * 2048) + (angle % 2048)
```

University of Kentucky

Conclusion

- **Full HW/SW design will be public domain**
  - 1st prototype was **Lafodis 4x5**, rectangular with two rails
  - **Lafodis160** is 2nd prototype, cylinder direct + rail
  - 3rd prototype will be cylinder herringbone gear + rail

- Links in paper & to be posted at:

Aggregate.Org

UNBRIDLED COMPUTING

University of Kentucky