Contents

[Projects 2](#_Toc159926872)

[Portable 5gallon grow 2](#_Toc159926873)

[Garage door exhaust 2](#_Toc159926874)

[Make a camera stand 2](#_Toc159926875)

[Design electronics soldering station based on cardboard or hardboard 2](#_Toc159926876)

[IoT Home Automation Parts 2](#_Toc159926877)

[Purpose 2](#_Toc159926878)

[Requirement Concerns 2](#_Toc159926879)

[System Design 3](#_Toc159926880)

[Strategy for simplifying provisioning and maintenance of devices 3](#_Toc159926881)

[Architecture 3](#_Toc159926882)

[Hardware/Firmware Modules 3](#_Toc159926883)

[Defined Projects 3](#_Toc159926884)

[Defined Modules for Projects - think “Adafruit Stemma”-ish 4](#_Toc159926885)

[Types of modules 4](#_Toc159926886)

[Design Considerations 5](#_Toc159926887)

[I2C for module interface? 5](#_Toc159926888)

[Schematic 5](#_Toc159926889)

[Board Layouts 5](#_Toc159926890)

[Fonts 6](#_Toc159926891)

[Part selection, connector type usage, pad size & shape 6](#_Toc159926892)

[Eagle Part Libraries 6](#_Toc159926893)

[Fusion 360 Board Layout Design Rule Check & Classes 7](#_Toc159926894)

[Combining modules into a single design 7](#_Toc159926895)

[Firmware – ESPHome 7](#_Toc159926896)

[Design artifacts 7](#_Toc159926897)

[Fusion 360 Project files 8](#_Toc159926898)

[PCB Fabrication options 8](#_Toc159926899)

[How to use modules to create single-sided DIY PCB creation at home 8](#_Toc159926900)

[How to import my eagle, or fusion files 8](#_Toc159926901)

[How to export design from Fusion 360 to your PCB manufacturer (JLCPCB in this case) 8](#_Toc159926902)

[Flashing ESP32 with ESPHome firmware 8](#_Toc159926903)

[Preparing the binary flash file using ESPHome on Home Assistant 8](#_Toc159926904)

[Boot first time (http://192.168.4.1) 8](#_Toc159926905)

[Reboot it , connect to ESPHome 8](#_Toc159926906)

[Communicating designs to consumers 9](#_Toc159926907)

[Flashy stuff 9](#_Toc159926908)

[Distribution ... flood the market with the designs, simple, one-click instructions for purchasing (with parts populated on the board) 9](#_Toc159926909)

# Projects

## Portable 5gallon grow

Make 5 gallon bucket with grow LEDs mounted, circulation pump, handle on the top as wooden bar.

## Garage door exhaust

<https://youtube.com/shorts/_5AndfMppTY?si=Nuef1Vaj-9rlH_Dz>

## Make a camera stand

* precisely parallel to table
* 25” above 8.5x11 sheet of paper on 3x magnification
* level setting adjustment
* LEDs around inside
* height adjustment
* fine layout grid on table with ruler
* wired and wireless remote shutter button (Bluetooth to phone) - casts live camera image to a separate cell phone, monitor or TV - Chromecast device

## Design electronics soldering station based on cardboard or hardboard

* Solder fume fan/extractor for office
* Wood table top for poles in garage with casters for electronics workstation
* Ceramic soldering iron sleeve

# IoT Home Automation Parts

## The problem –manageable home automation for the average guy

Here are some problems with existing choices available for home automation that contribute to barriers to entry and are intrusive into the daily routine of the technical consumer wanting to build a home automation system to simplify their lives at home on a long-term basis. The intended audience for this information is intended to be a “technically adept prosumer” that will need to be involved in combining technologies for the purpose of automating their homes.

1. ***No ONE manufacturer/standard – too many choices***
   1. Companies - Samsung, Apple, Wink, Phillips, Tuya, Nest (Google), SonOff, IKEA, Roku, …
   2. Protocols – WiFi\*, Zigbee, ZWave, Thread, Matter
   3. Firmwares - Closed-Source (Phillips Hue, Nest, …), ESPHome, Tasmota, Wink,
   4. Cloud and App providers – Google Home, Apple Homekit, SmartThings, Amazon Alexa, IFTTT,
2. ***Product lifespan is tenuous and short-lived***
   1. Caused by evolving standards, silicon innovation, …
   2. Companies are bought, sold and product lines retired along with support and documentation
3. ***Closed-Source products***
   1. Platforms are designed to serve the bottom line of companies, and to keep users locked into those platforms
      1. Poorly architected for general case homes (assume specific platforms like Windows, Apple OSes, network architectures, provisioning mechanisms that are hidden from home network admin)
      2. Cloud-based (open to hackers and prying eyes – less secure by design)
   2. Inter-node collaboration is almost non-existent, or not as customizable via code or config changes

## The solution

***THEStack - ESP32 + PlatformIO + ESPHome + Home Assistant + Docker***

***Pick One & Simplify*** - The following will address the problems with existing home automation solutions by reducing variables and choosing best-in-class components, platforms, firmware and protocols that can simplify the work involved in setting up and maintaining the nodes and their automations. In this case, that means choosing ESP32 node devices that can run ESPHome so they can all be setup, managed and communicated to the same way using Home Assistant.

***What happens if I get hit by a bus***? …or even sell my house with the home automation remaining.   
Who will be able to figure out how to maintain my home automation system? Simplifying the job of maintenance is the main reason to prioritize ESPHome-based nodes when possible, as they are easier to setup, backup, maintain and document for the person who takes it over with Home Assistant.

***Longevity*** – ESPHome, ESP32 microprocessors, Home Assistant and WiFi are not going anywhere for the foreseeable future – probably 😉

***Not Exclusive*** - But, what about new gadgets? Even with the introduction of standards like Thread and Matter, the above concerns are not solved or simplified to the point your grandma could create and manage home automation, but we can lessen the impact by simplifying our choices without scarifying functionality. There still may be some effort required to craft an automation solution using these choices, but the purpose here is to simplify and make those efforts more deterministic and reliable so they “just work” and won’t require a different app for each. Home Assistant can still manage all manner of devices and networking protocols, so other manufacturer’s devices are an option, but ESPHome becomes a consistent and capable choice when possible.

***Manage Complexity*** - There are, and will continue to be a wide variety of vendors, networks and platforms to choose from. The approach, strategies and tactics outlined here will not eliminate the use of any specific product, vendor, network or software, but to create a way to choose what will be the least impactful to creation and maintenance of a well-defined, and cooperative system that does not require constant learning and fidgeting with specific nodes or cloud software providers.

1. ***Future-looking solutions***
   1. Open-source platform – Home Assistant
   2. Large user base & comprehensive documentation
   3. Constant innovation - happens in response to need by talented users
   4. Excellent software architecture – flexible, scalable and as simple as possible
   5. Lessen impact of corporate product line fuckery
2. ***Choice of dominant node network protocol***
   1. WiFi 2.4GHz – widest availability in the average home – cheap WiFi access points are easy to add for better coverage without technical know-how being an important factor.
3. ***Granular control over combining functionality*** – Hardware and Software modules and common design patterns
   1. Modules & Design patterns
4. ***Consistent setup*** - establish new node, provisioning, monitoring, reporting and tracking over time
   1. Loading node initial firmware – ESPHome (USB cables)
   2. Loading node new firmware OTA
   3. Configuring node software (network credentials/secrets, parameters, logic)
5. ***Excellent architecture***
   1. Consistent power solution
   2. Consistent processor(s) – save $, flexible, runs ESPHome, enough GPIO, implements all these requirements, widely available going forward, …
6. ***NO Cloud*** – Secure LOCAL-ONLY communications by default with option for internet access if needed
   1. No recurring user fees
   2. Nobody able to see your passwords and activity such as when you are home
7. ***ONE App*** – one user interface

HOW – Requirements to satisfy the solution

Design

The following requirements are meant to provide consistency to the creation of a “family” of IoT printed circuit board modules that have a common look-n-feel, features, documentation and locations in project file folders.

The modules fit together with custom circuitry and features to implement IoT home automation projects that can be used together on a Home Assistant installation using ESPHome on a WiFi network as a common platform.

## Requirement Concerns

Home Automation addresses the following concerns that can be automated:

* Security & Privacy
  + Locking, unlocking and monitoring doors, windows
  + Opening and closing window blinds and shades based on:
    - Time of day
    - Ambient light
    - Use of television/theatre
* Lighting automation
  + Turning on, off and monitoring the status of lights
  + Switching on/off room lights based on:
    - human presence
    - time of day
    - ambient light available
* Health Support
  + Turn on/off exhaust fans based on:
    - Air quality
    - Humidity
    - Shower activity
  + Raise/Lower HVAC temperature:
    - On a schedule
    - When all people leave or arrive back home
  + Raise/Lower heaters for toilet seats, blankets
* Monitor to avoid property damage or loss of money
  + Water leak detection
  + Smoke detection
  + Pipe freezing
  + Energy usage
* UI – provide convenient home automation system user interfaces:
  + On mounted or portable tablets
  + On android TV
  + Via web browser on phone
  + Control automations via voice commands through room monitoring devices
  + Event notification – appliance needs attention

## System Design

### Strategy for simplifying provisioning and maintenance of devices

Open Source hardware and software

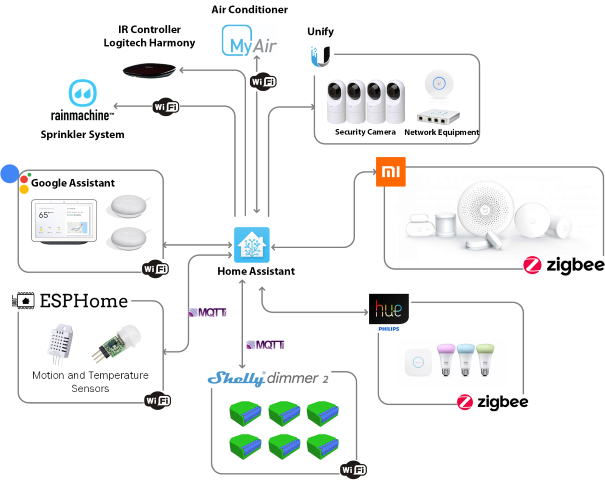
ESPHome

### Architecture

* Dedicated VLAN on home network to isolate home automation devices and software from the rest of the network and the internet
* Choice of wireless and wired network infrastructure
  + WiFi – 2.4GHz - < WiFi 5
  + WiFi 6, Thread, Zigbee, ZWave

### Hardware/Firmware Modules

Create home automation hardware/software modules to be reused for TKM’s home for purposes like:



Architectural Overview Drawing

## Defined Projects

|  |
| --- |
| TheCoffeeWarmer - |
| THEToiletSeatWarmer |
| THEElectricBlanket |
| THEBathroomScaleMonitor |
| THEOutsideCamera |
| Room Monitors |
| Bed Presence Monitor |
| Blinds Openers |
| Door Monitors |
| Window Monitors |
| BaseboardLED Strips |
| GarageBreakerMonitors |
| GarageLeakDetector |
| Water Usage |
| Fire Alarm Monitor |

## Defined Modules for Projects - think “Adafruit Stemma”-ish

***Goal*** - Single-purpose circuits and devices commonly used together in multiple projects encapsulated in its own hardware modules (e.g. 5V AC/DC power supply) – like a library of functional circuit designs with pre-chosen ESP32/ESPHome friendly parts

* + Facilitate simplified circuit ***reuse*** with consistent results across projects
  + Modules can be inserted into other designs, or used as an independent module on its own PCB ([think Adafruit’s Stemma QT](https://learn.adafruit.com/introducing-adafruit-stemma-qt/what-is-stemma))
  + simpler integration into projects using common headers representing input, output connections to the circuit

### Types of modules

* Sensor
* Monitor - track/report status/value of:
  + a binary switch: opened/closed, on/off
  + Temperature
* MCU

|  |
| --- |
| THEACDC5V2APowerSupply –   * LED power indicator on power supply module (add header to output LED externally) |
| THEESP32MCU-xAnt |
| THEESP32MCU-Lw-Pwr |
| THEInlineRelayModule   * Create over-current cutoff feature to the relay for drain circuit - add pins to a output header for outputting current measurement. (analog or digital?) * Add current monitoring circuit to the power supply * Have ability to limit current draw via current sensing module |
| THEBatteryModule |
| THEPresenceMonitor |
| THECameraModule |
| THELoadCellModule |
| THEDoorWindowSensor |
| THEEnergyMonitor |
| THEBlindsOpener |
| TheShadesOpener |
| THEBedOccupancySensor |
| THEWaterUsageMonitor |
| THELeakDetector |
| THEWLED |
| THEFireAlarmMonitor |
| THELightDimmer |
| Water flow volume/speed sensor |
| Light sensor |
| LED strip |
| \* Pump switch and/or motor speed controller |
| \* encoder sensor (measure rotational or linear movement and location) - fan speed, windmill rotation |
| \* air quality sensor (co2, dust, |
| \* Create breaker current monitoring sensor:  esphome + ESP8266? OR cheapest esp32 |
| * NFC Reader |
| * Appliance |
|  |

## Design Considerations

Automation should be engineered to blend with user’s day-to-day behavior’s to avoid the people in the home from doing anything, if possible.

Hardware module design conventions here can improve organization, simplify working with your designs, create a consistent look-n-feel to your designs or improve board robustness so they just work more often than not.

Logging

* Types of code debugging:
* serial
* mqtt
* log file
* syslog over network

### I2C for module interface?

Use I2C for the intra-module communication to MCU module (in parallel to esphome recording? ) :

\* voltage

\* Temperature

My PCB Design Checklist???

Testpoints, programming header, boot and reset buttons

With KiCad – use Symbols as global, and footprints as local to the project (with footprint libraries stored relative to the kicad project files. Symbols don’t behave in project specific ways, but footprints variants can be project specific – here’s how:

\*

\*

### Schematic

* + document power draw for each component on the schematic

### Board Layouts

***Goal*** - Create highly serviceable boards, especially for Silkscreen elements that provide integrational instructions (like header pin labels) for the integrator or the person populating the boards with devices.

* to help debug and place components - will help with placing components, debugging functionality or inferring functionality
* Attach attributes to components for identification of part numbers (for easier BOM creation that PCB manufacturers will use without a lot of back-n-forth)
* Draw 3mm diameter mounting holes in Eagle’s “Milling” layer (for JLCPCB) with a surrounding 5mm copper free area
* Intra-board connections with other circuits or modules should be consolidated through headers as input(s) or output(s). Place headers on opposite ends of the board (preferably from left-to-right like the schematic
* Create polygon “pours” for ground and power signals
  + Ground – bottom layer on 2-layer boards, 3rd layer on 4-layer boards to minimize impedance issues
  + Power – top layer on 2-layer boards, 2nd layer on 4-layer boards
* Leave larger traces uncovered by soldermask if they are to be flooded with extra solder to improve capacitance (eg. with AC traces – especially if those traces are thinner than they should be)
* layout board like schematic circuit

#### Silkscreen conventions

Silkscreen acts as guidance for the person placing the components. Leave no room for ambiguity, since I am the one hand-soldering the parts…see <https://www.nexpcb.com/blog/pcb-silkscreen-design-tips>

Helps identify components, variants, polarity as well as rating values.

* Shoot for a design that is free of clutter
* Reading direction - left to right, top to bottom:

Chip label directions:

* + Horizontal - upper left
  + Vertical - bottom-left
* Avoid Silkscreen over high-speed traces to avoid impedance issues
* Clearly mark Silkscreen with proper connector orientation
* Add a text element to the board layout on the silkscreen layer with the text “>DRAWING\_NAME”
* If the space allows it, you can add your company logo, tracking barcodes, QR codes, anti-static markings, etc..
* Add warning signs for exposed electrically harmful spots

### Fonts

***Goal*** - Use consistent fonts between schematics, board layouts, projects and Eagle libraries, simplified circuit inspection, tracing and debugging

* + Orienting labels in all the same direction to improve readability
  + Drawing element types of a schematic, board layout and library part footprints share the same font and font size
  + Makes it easier to figure out what you are looking at in the drawing or on the board.
  + The characters to be printed as the silkscreen should have a character line width to character height ratio of 1:5. Inter-character spacing should be greater than 5mil. The commonly used character line sizes are:
  + 4/20mil
  + 5/25mil
  + 6/30mil
  + For TKM designs:

*To see how fonts will be rendered by board fabricators, try out board fonts by exporting gerber files to manufacturer’s site and use their gerber viewer to see what they will appear. Remember that smaller fancy fonts might not look right when painted on the board (SIMPLE FONTs scale down better) – see* [*this link*](https://www.autodesk.com/support/technical/article/caas/tsarticles/ts/mUNCp0xBxwO1rJ9L5DOhL.html#25)

#### Schematic Fonts

|  |  |  |  |
| --- | --- | --- | --- |
| ***Schematic Drawing Element*** | ***Font*** | ***Target Schematic Layer*** | ***Size*** |
| Part Name attribute | proportional | 95 Names | 0.07 inch |
| Part Value | proportional | 96 Values | 0.05 inch |
| Part Number | proportional | 96 Values | 0.05 inch |
| Part Attribute | proportional | 96 Values | 0.05 inch |
| Text Value (note) | vector | 91 Nets | 0.07 inch |
| Text Value (descriptive text) | custom – Comic Sans MS | 91 Nets | 0.15 inch |
| Wire Name Label | proportional | 95 Names | 0.07 inch |

#### Board Layout Fonts

|  |  |  |  |
| --- | --- | --- | --- |
| ***Drawing Element*** | ***Font*** | ***Target Board Layer*** | ***Size*** |
| Part Name | proportional | 25 Names Top | 50 mil |
| Text signal Labels | custom - Bell Centennial | 21 Silkscreen Top | 32 mil |
| Part Number | custom – OCR A Extended | 27 Values Top | 50 mil |
| Identifier Text Labels (high level grouping of components or instructions) | custom - Comic Sans MS | 21 Silkscreen Top | 50 mil |
| Module Name | custom - Comic Sans MS | 21 Silkscreen Top | 50 mil |



### Part selection, connector type usage, pad size & shape

***Goal***:

Use similar parts for common use-cases

***Purpose***:

Make it easier to procure and select headers and board-to-wire connectors based on the wattage or frequency of a signal

* use spade connectors as an AC input option or 5.08mm pitch screw terminals
* use the new 2.54mm pitch female pins for programming or testpoints? Use exposed (larger?) vias instead of pads for pogo pins on test points and programming pins.
* use 5.08mm pitch connectors (blue, green,…) for screw terminals, headers
* use JST connectors for circuit inputs and outputs
* try to use one input, one output per module where possible to simplify integrations of the module
* use LONG shaped pads with screw terminal block and other through-hole parts (for easy soldering)

#### AC fuse sizing steps

* Determine the maximum fuse rating in amperages by applying this formula: P (Watts) ÷ V (Voltage) = I (Amps)
* Calculate the minimum fuse rating or amperage by multiplying that total (the maximum amperage rating) by 125%
* Find out the wattage consumed by the equipment, divide by the voltage supplied, which will give you the amperage used by the device
* Round up to the next fuse rating available
* A rule of thumb is to derate a fuse by 25% to 50%
* For example, if a circuit has a continuous current draw of 12 Amps, then a fuse of between 16 Amps (25% derating) and 24 Amps (50% derating) might be selected

#### MCU Choice

ESP32-S3

ESP32-C3 – low power

ESP32-H3

Why the esp32 module(s) i chose - already fabricated and works, can be populated with header pins

* Don't have a reliable version of my own yet

### Eagle Part Libraries

* Library name – ***TKM-IoT-PartLibrary***
* File storage – exported library (with 3D models) located:
  + Local hard drive ***Documents\THEHome\eagleLibraries\TKM-IoT-PartLibrary.lbr***
  + In Fusion 360 projects - ***MyParts\TKM-THEIoTComponents\libraries*** folder

#### Consistent Part Naming

***Goal:***

* Create consistent Schematic and Board Part Name Identifiers

***Purpose:***

* Make it easy to look at all my designs, and figure out how everything is put together across all modules or projects

|  |  |
| --- | --- |
| ***Part Name ID*** | ***Component Name ID*** |
| Capacitor | C$1 |
| Diode | D$1 |
| Fuse | F$1 |
| Header | H$1 |
| LED | L$1 |
| Resistor | R$1 |
| Transistor | T$1 |
| IC/Module | U$1 |
| Wire Name IDs | prefix by ‘***AC\_***’ for alternating current wires |

### Fusion 360 Board Layout Design Rule Check & Classes

***Goal***:

* One time configuration of Fusion 360’s CAM processor rules according to manufacturer’s capabilities

***Purpose***:

* Consistent trace widths by “class” by NET signal type

e.g. JLCPCB 2 and 4-layer rules

fabrication instructions :

\* how-to single sided DIY pcb creation at home - include zero-ohm resistors (wires)

\* how to combine modules into a single pcb

\* how to export design to JLPCB in Fusion 360

\* how to import my eagle, or fusion files

\* animation of modules flying into a circling ring around the TKM logo extruded to 3d photrealistic steel gear

\* how to choose parts

\* why the esp32 module(s) i chose - already fabricated and works, can be populated with header pins

- don't have a reliable version of my own yet

Final step... flood the market with the designs, simple, one-click instructions for purchasing (with parts populated on the board)

\* github pages that say, as-is with THEIOTMODULES@TKM.COM

\* ON fabricators sites, jlcpcb pcbway, oshpark?, ...

### Combining modules into a single design

### Firmware – ESPHome

Use ESPHOME features that allow for offline automations (AND automation with other ESPHome Devices???)

Folder structure on Home Assistant

commonBoards

commonSensors

Yaml file structure

## Design artifacts

* Export module designs from fusion MyParts\TKM-THEIoTComponents folder to Documents\THEHome\components
* create each module in a separate electronics design
* named **THE***{boardpurpose}***IoTModuleXY** (eg. **THERelayIoTModule**)

where **X** represents a ESP32 series designator:

* esp32-s3 = S (XIAO)
* esp32-c3 = C (ADAFRUIT)
* esp32-cheapest... = 1 (sparkfun)
* lilygo display = D (touch display)

where **Y** represents the “module type”:

### Fusion 360 Project files

#### Module Designs

#### IoT Projects

##### Schematics

##### Board Layouts

## PCB Fabrication options

* + cover exposed copper with tinning solution and solder and eventually with clear fingernail polish or soldermask for home PCB creation
  + make polarity obvious - the biggest reason for assembly failures
  + try to orient like-polarized components (capacitors, diodes) the same to avoid placement mistakes

### How to use modules to create single-sided DIY PCB creation at home

* + - Use zero-ohm resistors (wires) to bridge traces normally done on different layers
    - Use custom CAM jobs using the CAM processor to generate files necessary for etching via photoresist chemical process
    - Positive or negative …

### How to import my eagle, or fusion files

### How to export design from Fusion 360 to your PCB manufacturer (JLCPCB in this case)

## Flashing ESP32 with ESPHome firmware

### Preparing the binary flash file using ESPHome on Home Assistant

#### Connecting via wires using adapter??? Or FTDI…

Main thing to understand is wiring…

* + - GPIO0 to \_\_\_\_
    - 3.3v
    - GND
    - TX
    - RX
    - ll

<https://hackaday.io/project/171449/components>

<https://hackaday.io/project/176199/instructions>

### Boot first time (<http://192.168.4.1>)

* Plug in the WIFI credentials (SSID and Passphrase for the 2.4GHz only network the device will communicate through)

### Reboot it , connect to ESPHome

## Communicating designs to consumers

### Flashy stuff

#### Animate modules flying into a circling ring around the TKM logo extruded to 3d photoresist steel gear

### Distribution ... flood the market with the designs, simple, one-click instructions for purchasing (with parts populated on the board)

* github pages that say, as-is with [THEIOTMODULES@TKM.COM](mailto:THEIOTMODULES@TKM.COM)
  + backups of docs, wiki, fusion 360 backups, ESPHome YAML/CFG files
* Add project to fabricators sites, jlcpcb pcbway, oshpark?, ...
* Wiki
* Issue tracker
* Feature backlog