Lecture 23

Range sensing and Wired Communications (I2C)

Today's Agenda

01. Range Sensing

02. Review Wired Communications [Overview]

02a. Synchronous Communication I2C

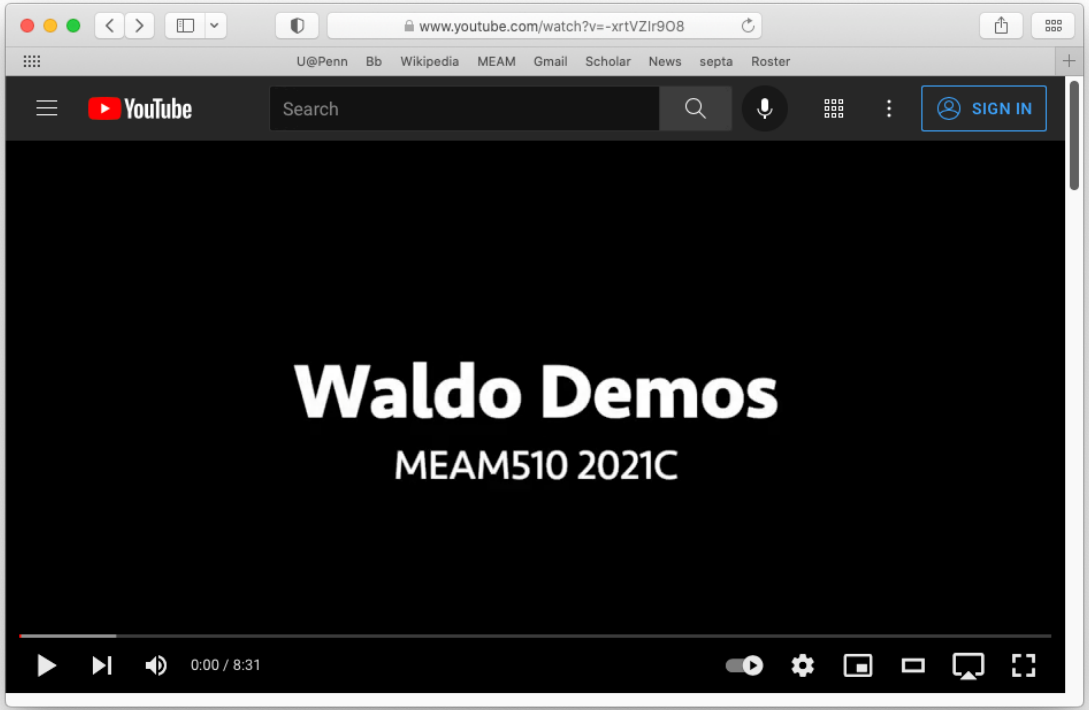
Stuff

- PD70-01C diodes have come in. Teams can get more than two if they want.
- More Deans' connectors in ministore.
- Backup ESP32's have arrived
  - Backwards Picokits (sockets instead of header pins) (only 8 units)
  - NodeMCU from last year (recycled) (dozens)
- Stronger servos available for purchase MG966R equiv.
- COVID reminder
  - The percentage of people in the hospital that are vaccinated is rising. (Used to be 1 in 10, some say now 1 in 3)
  - The average age of hospitalized COVID cases is getting lower.

Design Review – This Wednesday 3:30 on Zoom

- 15 min per team ~8 min presentation ~7 min feedback from coach and cohort
  - Give and take ideas from each other. It's not cheating…
- Presentation might include:
  - Ideas for the overall approach
  - Critical function – item you might have the most problem with.
  - Long lead time items (e.g. anything you plan to purchase).
  - How you plan to solve each task
  - Schedule for development including consideration for other classes.

Lab 3 Waldo Compilation Video



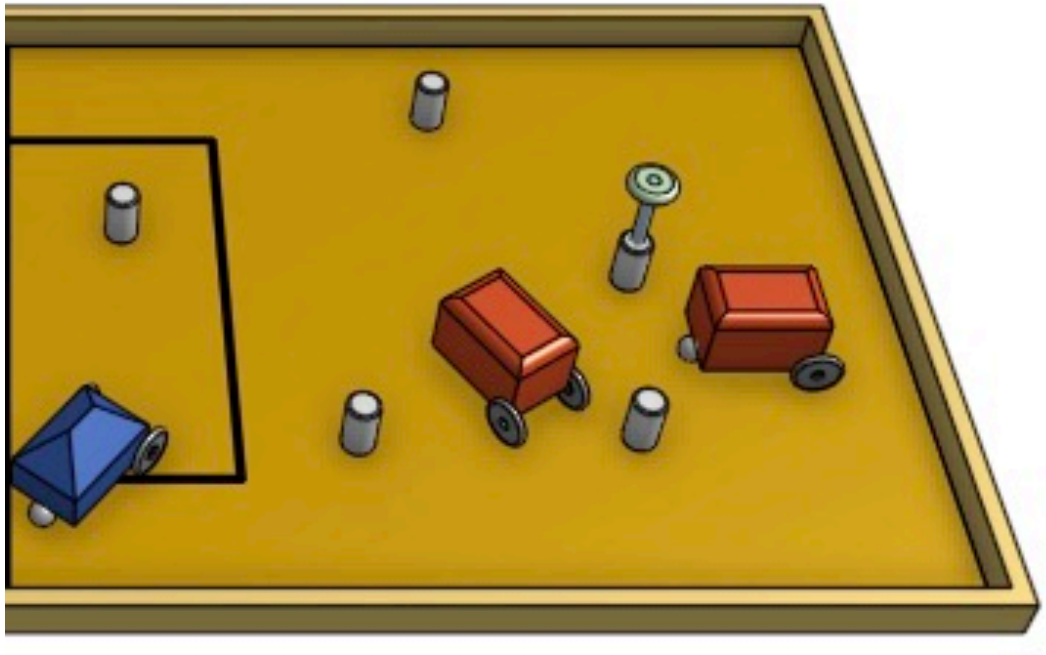
https://www.youtube.com/watch?v=-xrtVZIr9O8

01

Ranging Sensors

Ranging Sensors in 510 GTA game

- Vive position updates occur only once per second. Not enough to track a moving enemy robot.
- Maybe useful to know how far beacon is to some precision.
- Range sensors can be used to detect wall.



Things to think about for quiz at end:

1. You have a robot that is moving in a straight line and you want to accurately estimate when it will hit a wall (not using Vive)
2. You have $5 budget left and you want to sense if something is 2 cm in front of the robot so you can grab it.
3. You have $5 budget left and you want to sense if something is 50cm away to help navigation.
4. You have lots of budget left and want the most precise way of detecting if something is in the range of 50cm away

Ranging Technology
– measuring distance to objects

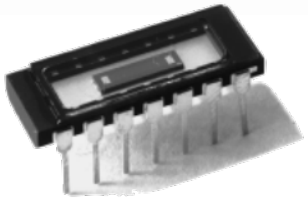
- Four mechanisms for measuring distance (as far as I know)
  - Time of flight
  - Intensity of reflection
  - Triangulation
  - Contact sensors (e.g. whisker)
- Forms of energy used
  - Light (e.g., IR proximity sensors)
  - Sound/vibration (contact, ultrasonic transducers)
  - Radio frequency EM (e.g., radar)
  - Heat (PIR, or pyro-electric) – human motion detection – not distance.

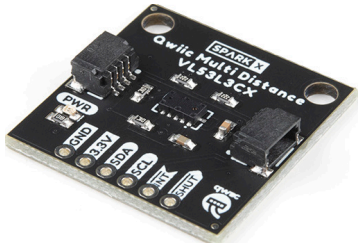
Ranging Sensors

- Ranging sensors determine the distance between objects.
- Typically energy is emitted and reflected energy is measured (intensity, location, time).
- Sometimes energy from environment is used (e.g., cameras)

Range sensors

- Proximity Ranging
  - ~$1-5 Cheap, short range, inaccurate, single point
- Ultrasonic range sensor
  - ~$4-50 Cheap, unreliable, single point
- Lateral effect photo-diode
  - ~$15-50 Fast, single point
- Triangulation (structure light vision)
  - ~$30-400, typ. Laser line stripe, and video cam.
- Patterned stereo vision (Kinect)
  - ~$200, cheap cameras, need large processing
- Lidar (sick™ hokuyo™)
  - ~$1000-3000, relatively big expensive
- Time of Flight (phase based) ranger
  - ~$10-15, small, slow, but relatively accurate





Choosing range sensors

- **Cost** (can be pennies to thousands of dollars)
- **Min/max Range** is always critical
- **Noise** (precision and accuracy)
  - Precise -> consistency in values
  - Accuracy -> correct distance measure
- **Sampling speed**
  - Probably not critical for final project
  - Can increase precision (by taking many samples and averaging)
- **Beam width**
  - Will increase precision of existance or absence of objects

IR retro-reflective

- Point ranging
- Intensity based
- Typ short range (~.01 to ~1m) dependent on source light intensity.
- Useful for detecting presence or motion
  - Hand under a water faucet
  - Person sitting on a toilet
- Color or reflectance of object changes reading
- Can get rough estimate of absolute range
- Lensing defines active sensed area
- Can be very cheap!

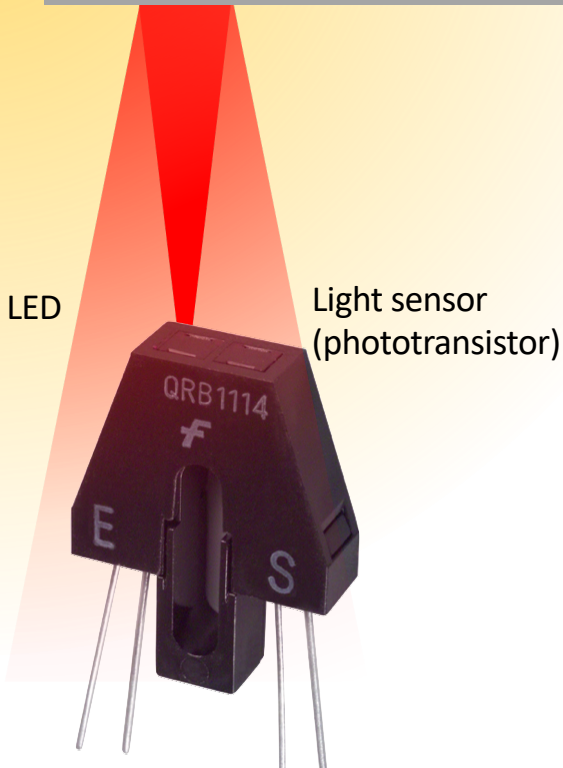
Q1 Can you build your own IR retro-reflector with parts you used in Lab 2?

Retro-reflection Operation

- Intensity of reflection sensed from LED light
  Roughly by inverse square of distance → $P = 1/R^2$
- Ambient light adds sensor offset
- Sensing with LED on and off can remove ambient offset.

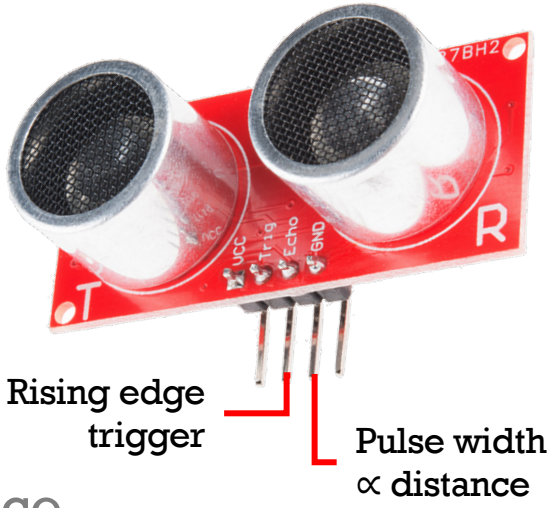
Q2 What can we do about compensating for the shift from the ambient light?

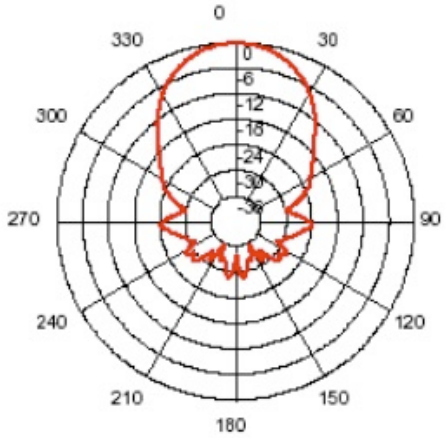
```
int  sensorOutput (){
  int ledsignal, ambient;

  digitalWrite(LEDPIN, HIGH); // turn LED on
  delay(1);                   // LTR4206 needs 10uS to respond
  ledsignal =  analogRead(SensorPin);
  digitalWrite(LEDPIN, LOW); // turn LED off
  delay(1);
  ambient =  analogRead(SensorPin);

  return (ledSignal - ambient);
}
```



Ultrasonic Ranging

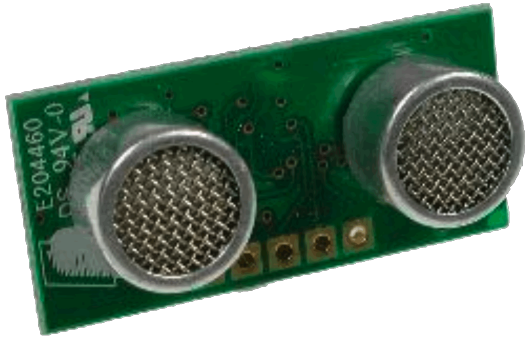
- Point ranging
- Time of flight based
  - speed of sound = 340m/s
  - E.g. 30m takes ~0.1s
- Uses timer (digital IO) to measure distance
- Range typ. (0.03m to 10m)
  - Cannot sense too close - typ transducers ring
  - Gain scheduling for better long range detection
- Sample rate limited by time for ping to dissipate
- Beam width varies
- Chirp typ. slew of freq, for better reflectance





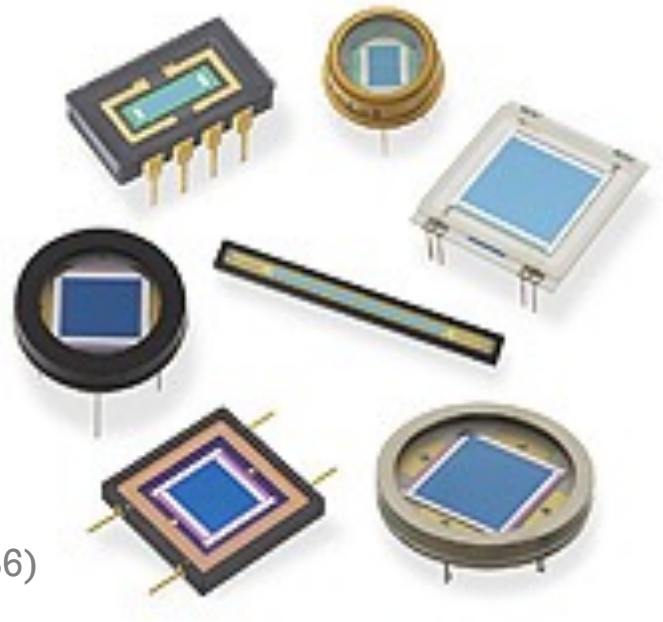
Example Ultrasonic Range Sensor

- RCWL-1601
- $3.95
- Range: 10cm – 250cm
- Speed: ~10 samples/sec
- Beam width: typ ~75°
- Sample Arduino code on Adafruit

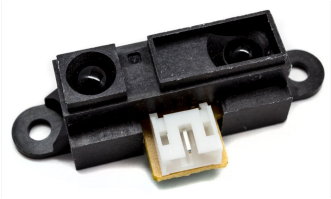


https://www.adafruit.com/product/4007

PSD: Position Sensing Device
[aka position sensitive detector]



- Point ranging via Triangulation
- Lateral effect photodiode
  - Returns the centroid of falling light
  - Insensitive to variance in amplitude
  - Can be precise (um sometimes nm)
  - Can be fast (typ > 100K samples)
  - Tend to be expensive compared to diodes (cheapest ~$6)
- Applications
  - Alignment motion axis of laser gyro
  - Bore sighting in autocollimator
  - Wheel alignment on cars
- Vendors
  - Hamamatsu
  - On-Trak Photonics Inc
  - Sharp – makes PSD packaged w/lenses

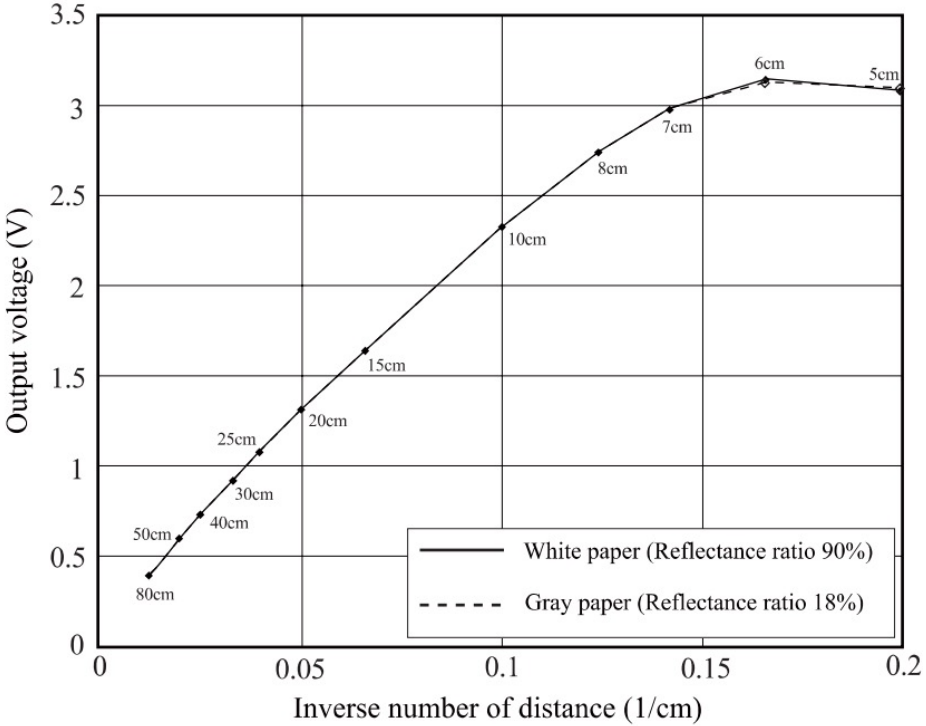


PSD based sensor units – packaged w/lenses

https://www.adafruit.com/product/164

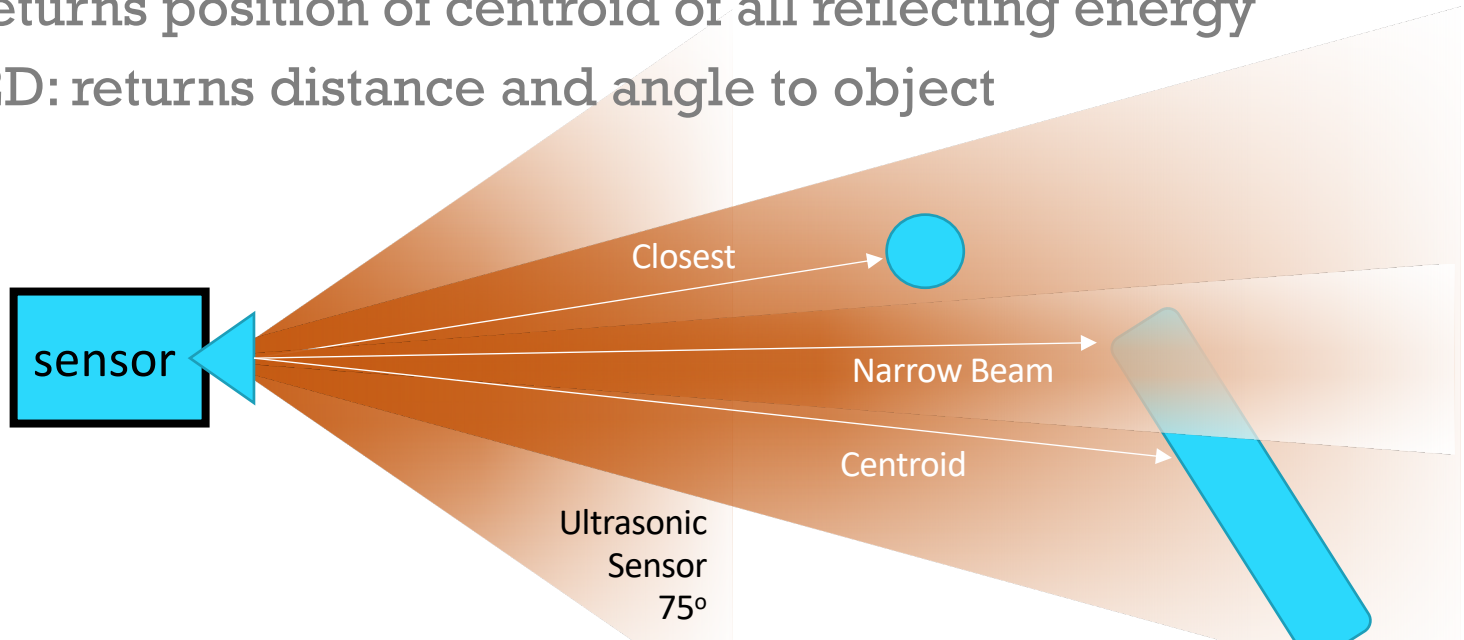
Sharp GP2Y0A21YK0F

- Range: ~6cm to 80cm
- Speed: ~20 samples/sec
- Beamwidth: 20-40 deg? (guess)
- Uses ADC
- Mostly independent of object color
- $14.95



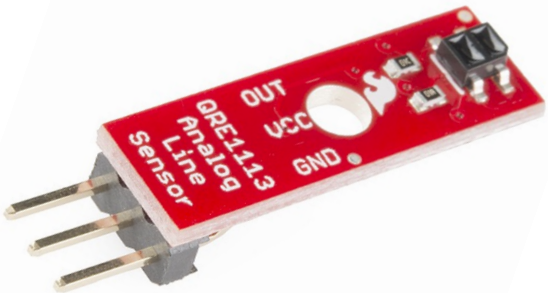
Beam Width

- Sensing anything in FoV of transmitter – reports distance to closest point that reflects enough energy
  - Smaller FOV gives more precise location
- PSD: returns position of centroid of all reflecting energy
- ZX is 2D: returns distance and angle to object



Black Line Detector (Redbot line follower)

- Short distance IR emitter/detector pair with integrated resistors. (Just retro-reflectors with resistors)
- Very simple, Uses ADC
- Range: ideal 0.125” from surface to white/black edge of line.
- ~$2.95, or build your own.
- Probably will work w/3.3V @ Vcc



https://www.sparkfun.com/products/11769

Flex sensor (whisker)

- Bending sensor changes resistance
- Simple to use with ADC in voltage divider.
- $8.95 to $12.95
- Can use like whisker for wall following
- Range: 2-4"
- Speed: as fast as ADC

https://www.adafruit.com/product/182
https://www.sparkfun.com/products/10264

Force Sensing Resistor (FSR)

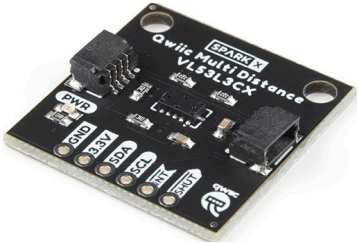
- Push on black pad, resistance changes
- Lots of hysteresis (relatively crappy sensor)
- Not very senseitive.
- $6.95, Some available in GMlab
- Use ADC in voltage divider.

https://www.sparkfun.com/products/9375

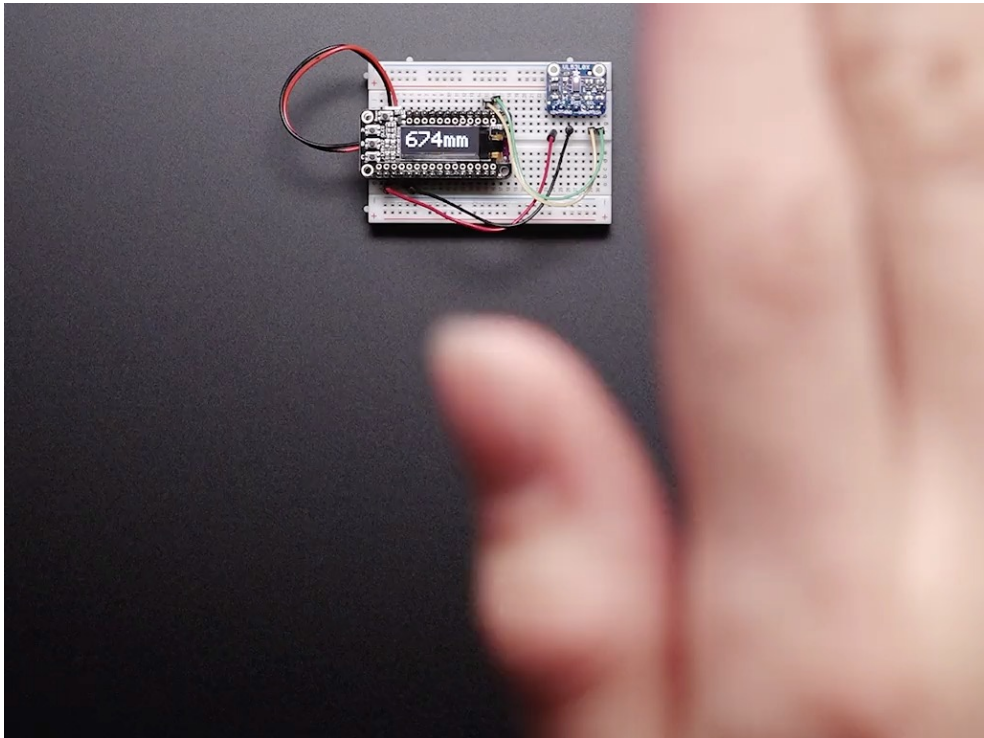
Time of flight and Lidar sensors

- Light Detection And Ranging (time of flight)
  - Sick
  - Hokuyo
- Functions
  - 2D scanning (line of points)
  - Okay to use outdoors
- Typical specifications
  - Typ range 0.1m – 30m
  - Scan rate ~50Hz
  - Resolution 0.25° @ 3-5mm
  - Price $2k to $6k depending on model
- Historically important to robotics (Solved SLAM)
- Single point sensor (TFMini-S micro LiDAR) $39.95

TOF Range Finder (VL53L0X)



- Range: 5cm to 120cm
- Speed: ~30 samples/sec
- Beam width: ~25°
- $14.95
- Interface with I2C,
- Adafruit has ESP32 compatible Arduino library.
- Very accurate and linear



https://www.adafruit.com/product/3317

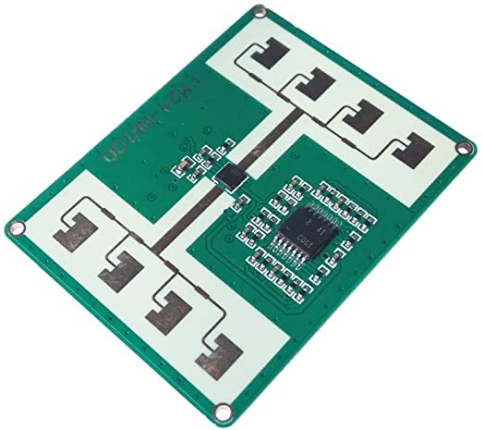
ZX sensor (2 IR emitter/detectors)

- Z-axis 12” range
- X-axis 6” range
- Speed: 50 samples/sec
- FOV: 2D sensor (centroid+distance)
- $26.50
- UART or I2C interface
- 3.3V ok
- Arduino code on sparkfun
- Relative motion is accurate, abs position is not
- Subject to color of object

https://www.sparkfun.com/products/13162

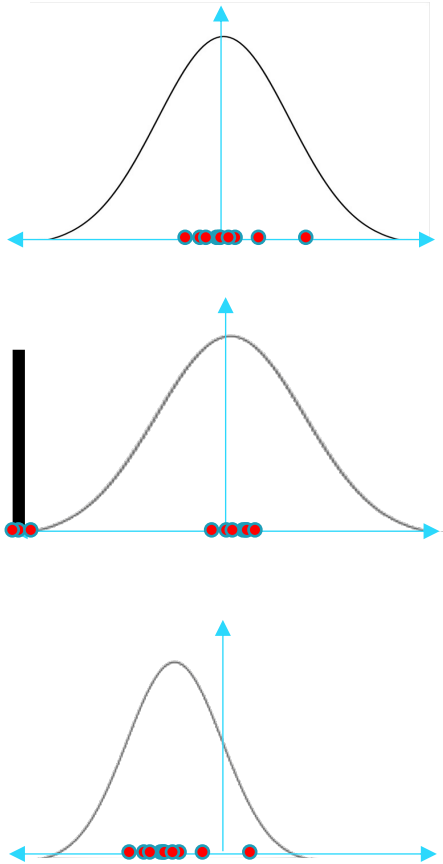
Radar sensors (interesting but not useful for 510)

- HB100 doppler sensor
  - ~$7
  - Not directional
  - No distance information
  - Detects can detect moving people or metal (sees through walls)
- FM24 NP100 FMCW ranging radar
  - Directional readings
  - Detects range to large conductive objects
    - People up to a few meters
    - Cars up to 10 meters
    - Sees through walls
  - $68 at aliexpress
    - https://www.aliexpress.com/item/32914896156.html



Noise characteristics – Precision vs Accuracy

- Variable data - gaussian distribution
  - If center of distribution is accurate, sensor can be accurate but have low precision (take average of many data points to increase accuracy)
  - TOF and PSD
- Unreliable data – bimodal
  - Occasional missed data. False negatives
  - Ultrasonic Rangers
- Distance susceptible to non-distance variables (e.g., object reflectivity, color)
  - Retro-reflectors, ZX sensor



Summary Range Finders

	Cost	Range	Speed	Beam width	Precision	Accuracy
Retroreflective	< $2	0 – 20cm	>500Hz	Custom	Med	Low
ZX retroreflective	26.50	0 – 80cm	50Hz	2D	Med	Low
Ultrasonic (ping)	3.95	2 – 450cm	~10Hz	~75°	Low	Med
PSD	14.95	10 – 80cm	~20Hz	Centroid	High	Med
TOF	14.95	5 – 120cm	~10Hz	25°	Med	High

Summary Quiz

Select which of the following range sensors is the most appropriate for each case:

a) Retroreflective
b) ZX retroreflective
c) Ultrasonic (ping)
d) PSD Position Sensing Device
e) TOF (Time of Flight)

1. Accurately sensing the time for a robot moving in a straight line to hit a wall.
2. Lowest cost way to sense anything 2 cm infront of me?
3. Lowest cost way to sense anything 50cm in front of me?
4. Most precise way to sense an object 50cm away

02

Wired Communication (Overview)

Overview (this and next lecture)

- Parallel Communication
  - GPIO state changes
- Synchronous Serial
  - I2C
  - SPI
  - (others)
- Asynchronous Serial
  - UART

Terms to understand by end of this lecture

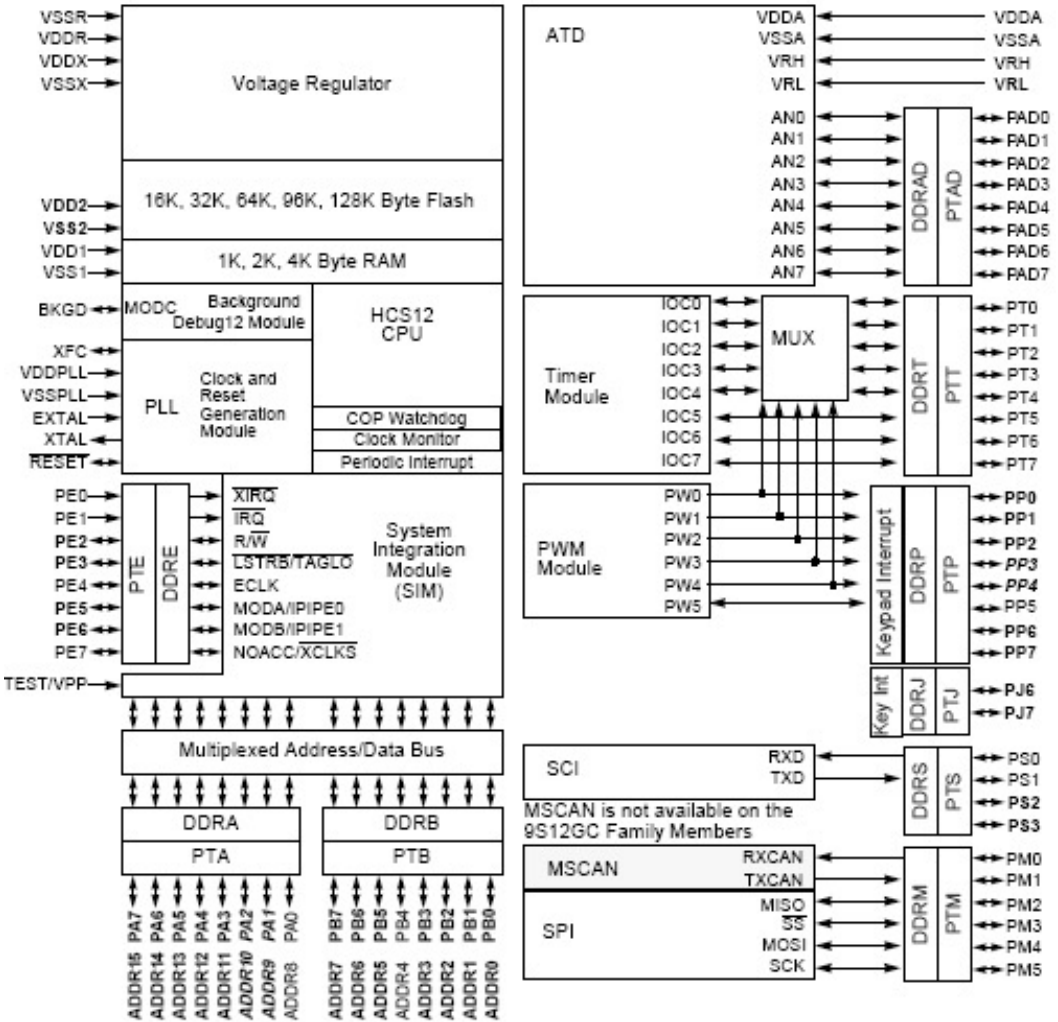
- Asynchronous vs Synchronous
- Parallel vs Serial
- Master-Slave vs Peer-Peer vs Client-Server
- Multidrop vs Point-to-point vs Broadcast
- Half Duplex vs Full Duplex

Quiz questions at end of next lecture

1. For asynchronous comm, you must set the same _____ on both ends
2. For synchronous, _____ sets the clock speed (though I2C has some exceptions to this)
3. _____and ____ are the two most common *synchronous* protocols used between embedded processors
4. _____ can be used for simple limited state changes

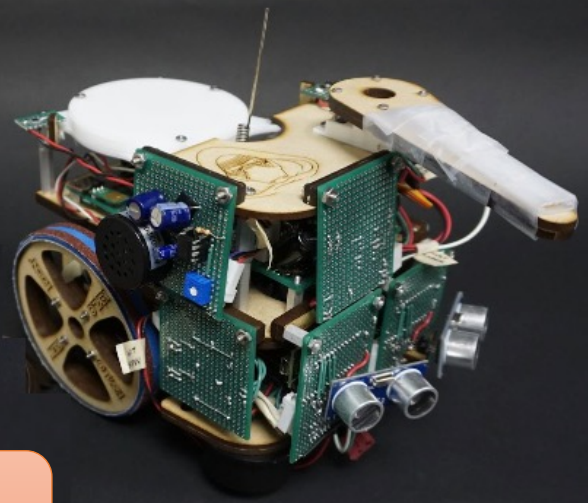
How do you choose which MCU(s)?

- Start with pin requirements and functions
  - How many GPIO?
  - ADC?
  - Timers? PWM?
  - Others?
- Allocate pins, decide how many MCU



Coordination between MCU's

Q3: Draw and hold one line that connects the microcomputers in each subsystem to indicate who needs to talk to whom



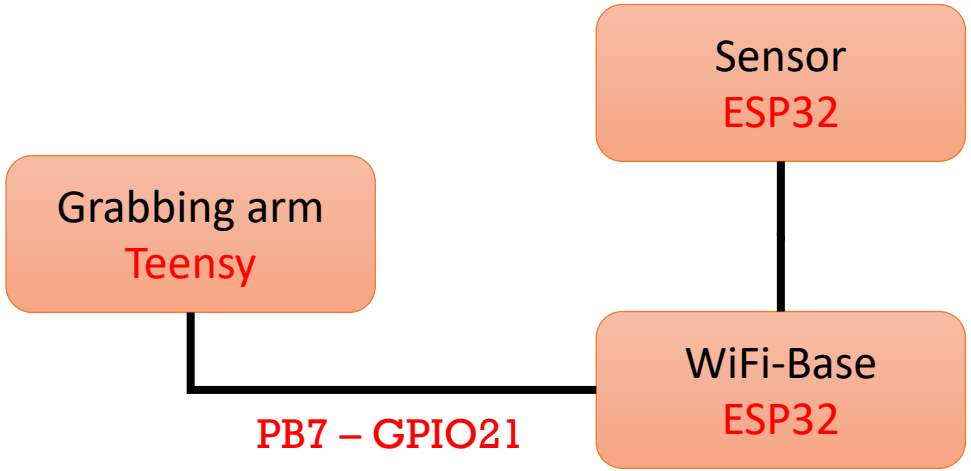
Sensor
ESP32

Grabbing arm
Teensy

WiFi-Base
ESP32

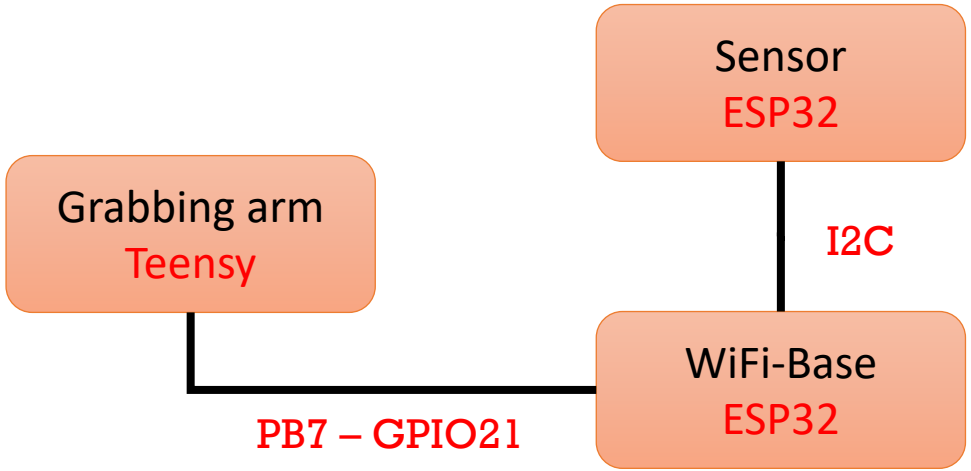
Coordination between MCU's

Q4: What is the simplest way for the base ESP32 to tell the arm Teensy to grab?



Coordination between MCU's

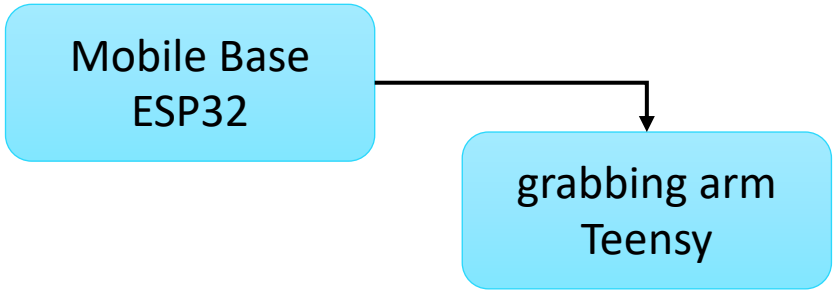
- Example networking of devices
- Usually constrained by subsystems and what pins are available.



Parallel communication

Q5: What signals could we send using two GPIO pins to the grabbing arm in order to do three things:

1. Close gripper,
2. Open gripper,
3. Lift arm holding closed gipper,



Q6: How many states could we control with three GPIO?

Serial Communications

Sequence of bits are used to send arbitrary states (messages) from one to another.

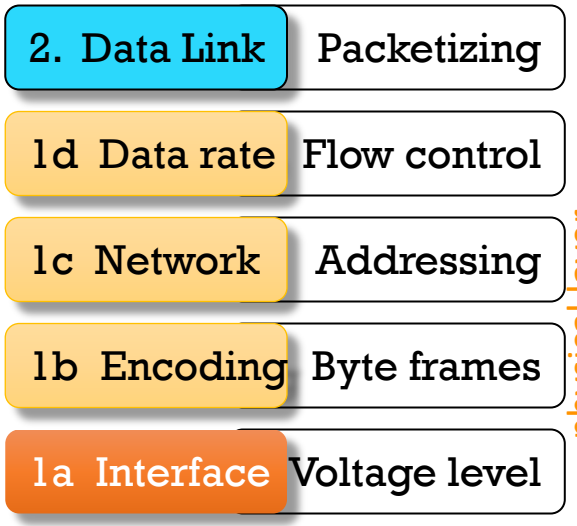
- Transmitter sends bytes (8-bits), transmits one bit at a time.
- Receiver gets bytes of data, reassembles bits into bytes.

Two ways to implement:

1. Hardware module
   - Specialized hardware to automatically assembles bytes without main software control. Runs in parallel like the timer subsystem.
2. Software serial ports
   - Main software drives GPIO (uses external drivers if needed)
   - This is sometimes called *bit-banging.*

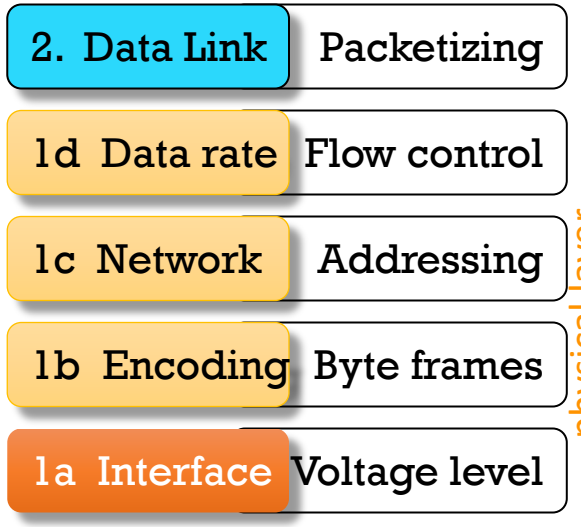
Communications between MCUs and peripherals

- Some protocols specify different components:
  a. **Interface** and voltage levels, sometimes connector physical specs.
  b. **Encoding** bit format and byte frames
  c. **Addressing** multiple devices/multiple masters
  d. **Flow control** (stopping or slowing data rate)
- Can be wired or wireless.



Communications between MCUs and peripherals

CAN	UART			SPI	I2C
	Logic	RS232	RS485		
x					
		o			x
x				x	x
x	x	x		x	x
x		x	x	x	x



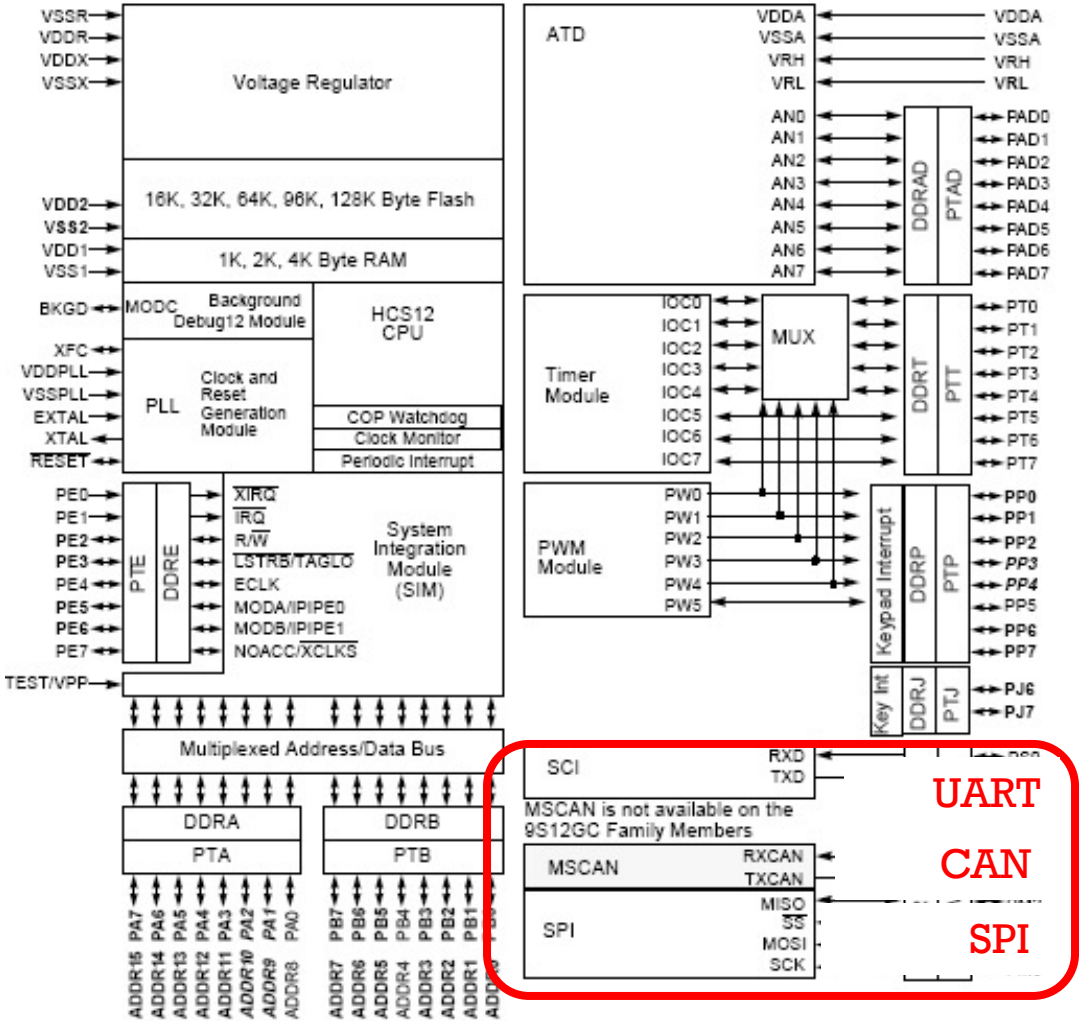
x: protocol specifies

o: protocol has this option (often omitted)

Motorola 9S12

- Similar to 68HC11 family

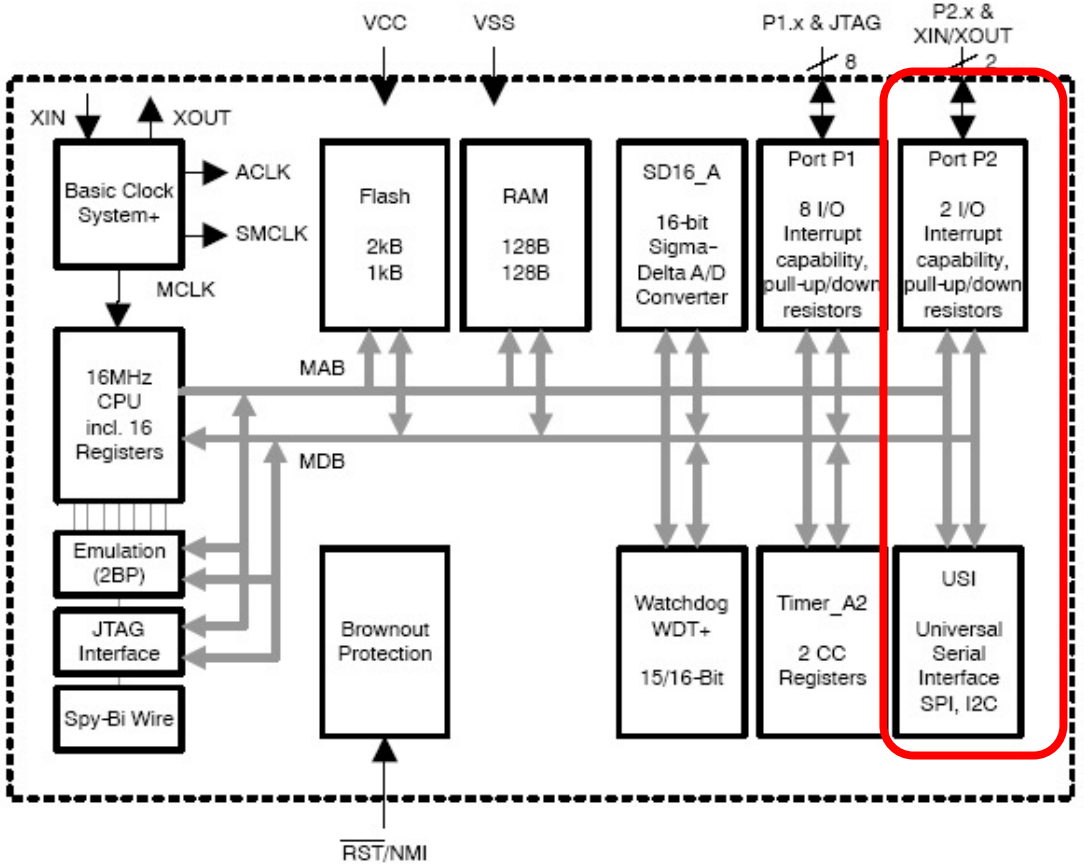
(used in MEAM510 fifteen years ago)



TI MSP430F2013

- Low cost RISC minimalistic

(used in MEAM510 ten years ago)

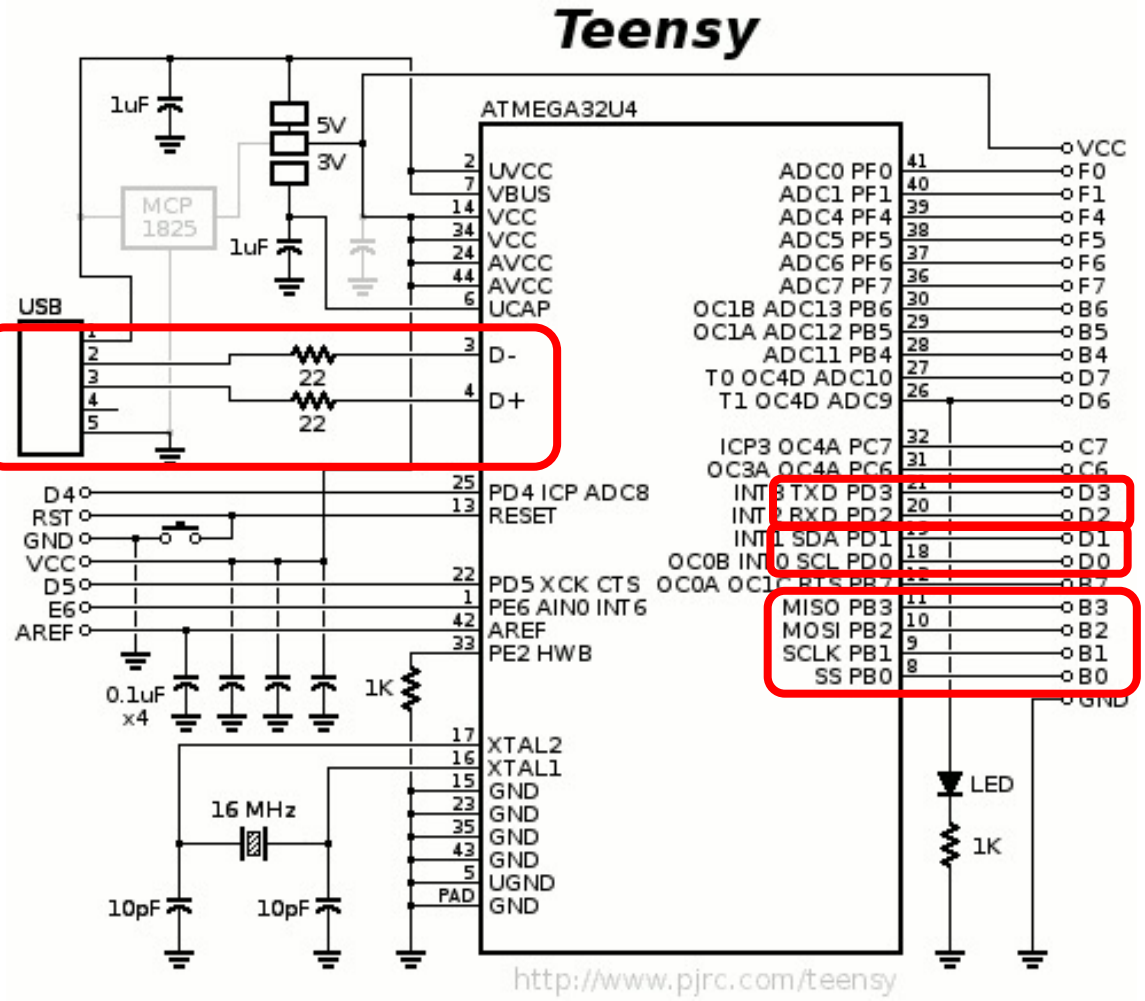


3 modes on same pins

UART

SPI
I2C

Teensy



USB

UART

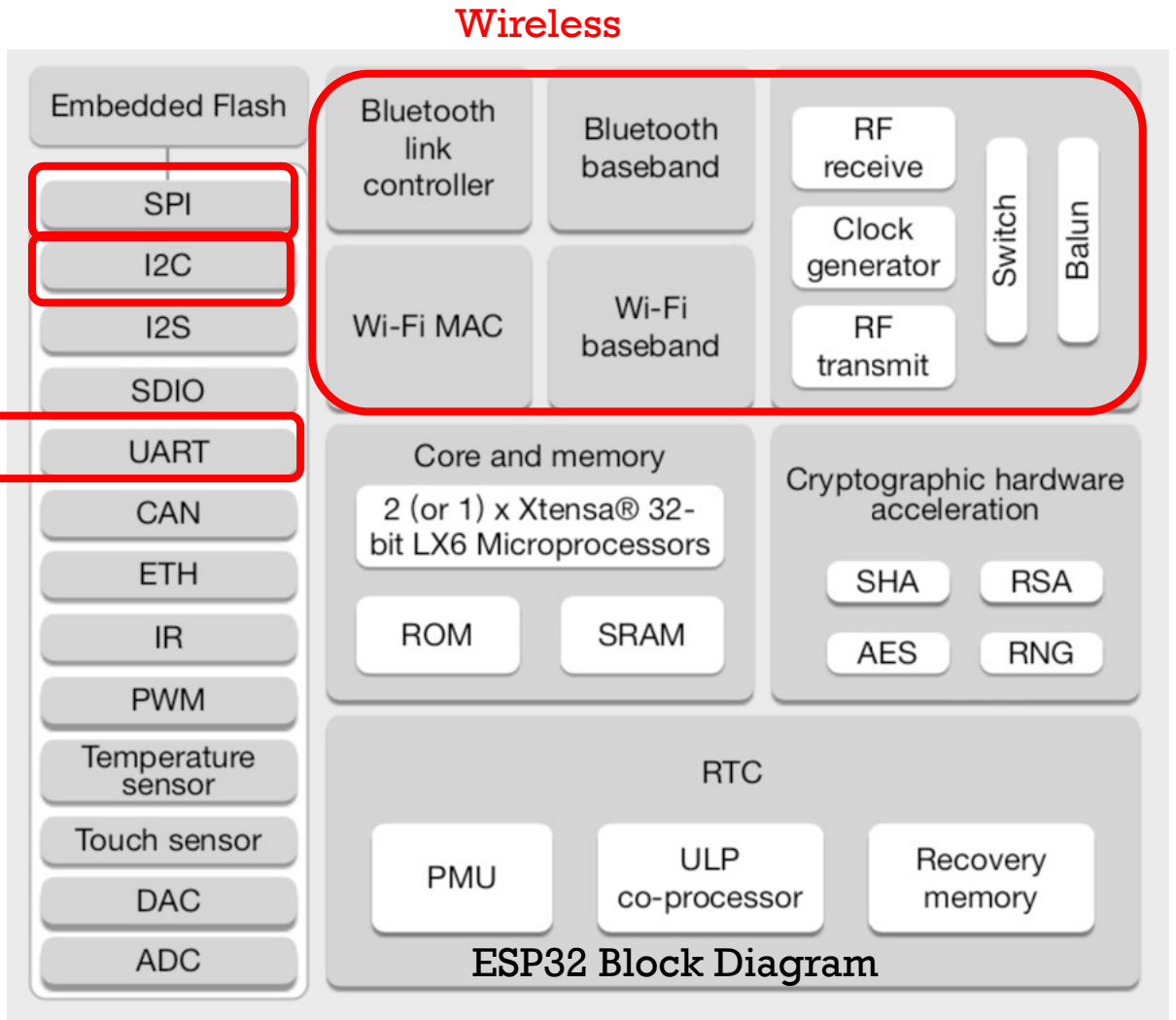
I2C

SPI

PicoKit ESP32

USB to UART

On PicoKit

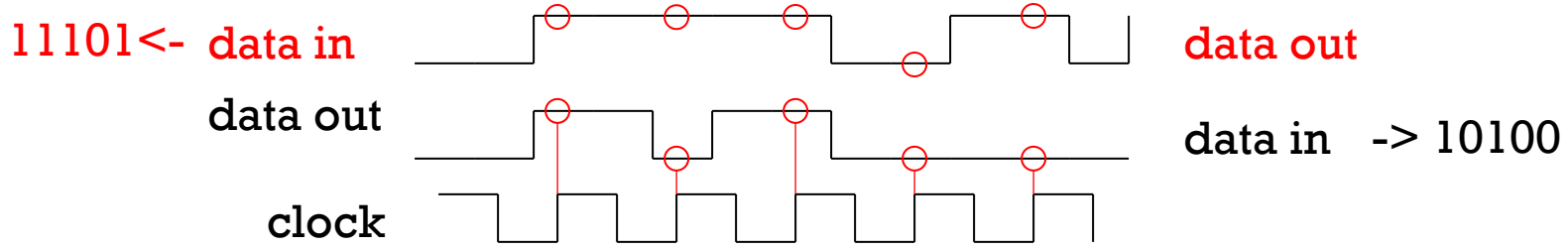


ESP32 Block Diagram

Asynchronous vs Synchronous, Half vs Full Duplex

Several characteristics of serial communications.

- *Synchronous* usually means there is a clock line that indicates when data is ready (e.g. each rising edge of the clock)



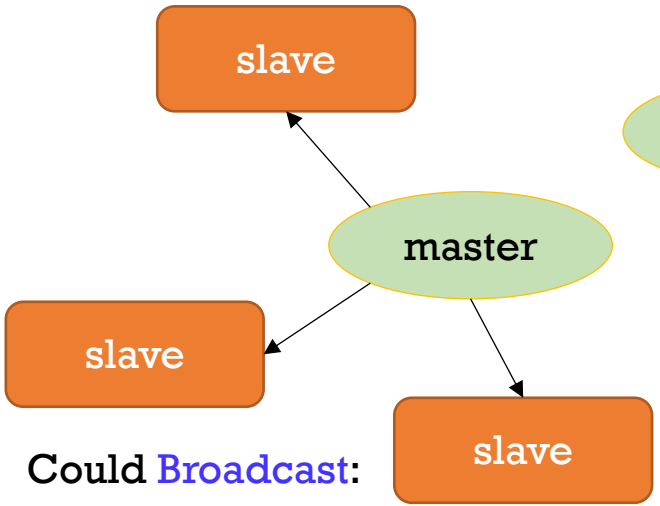
Half duplex for 1 way comm (can switch directions)

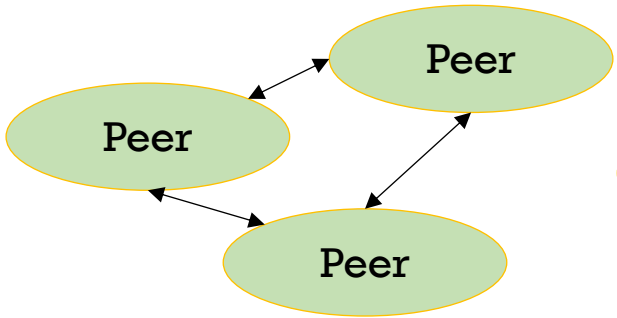
Full duplex for 2 way comm

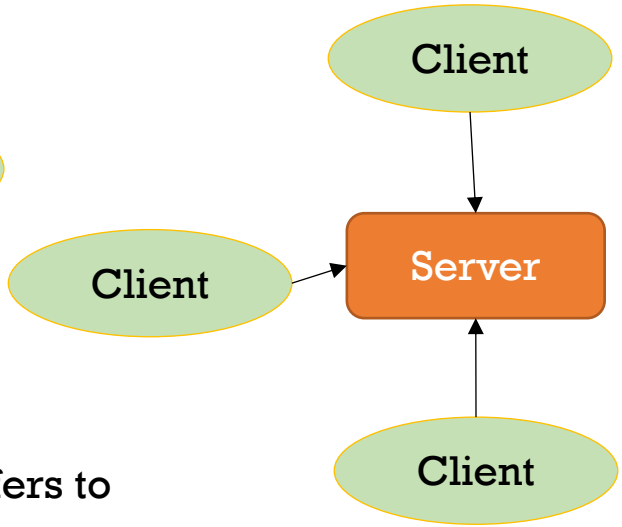
- *Asynchronous* usually means no clock line, receiver checks for line state at agreed upon clock rate.

Master-Slave vs Client-Server vs Peer-Peer

- Master and Client, initiate communication.
- Typically master-slave is many-slave to one-master. Client-server is many-client to one-server.
- Peer-to-Peer means anyone can initiate.







Could Broadcast: simultaneous to many

Point to Point in wired refers to one sender, one receiver

Multidrop refers to wiring connectivity, multiple devices on one line so addressing is required.

02a

Synchronous Comms

I^2C

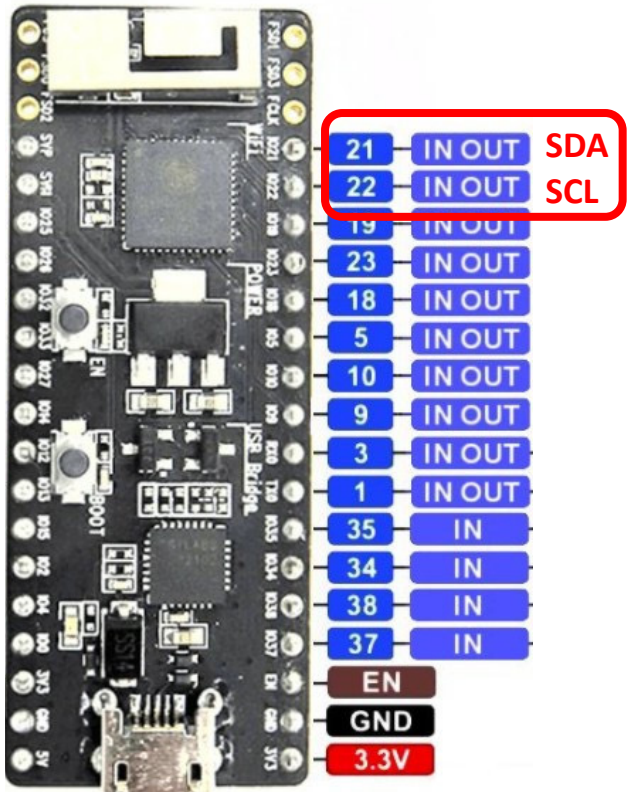
Synchronous Serial Communication

- Synchronous means that there is a line dedicated to indicate when data transfers between all senders and receivers occur.
- Usually this line is called the CLOCK line or CLK.
- SPI and I2C are the two most common synchronous communications protocols between two processors
- Both are very similar except:
  - I2C is half duplex. SPI is full duplex
  - I2C can use as few as 2 lines, SPI needs 4 lines, (possibly 3)
  - I2C max speed is slower than SPI max speed
  - I2C can be made lower cost than SPI
- We will cover SPI in the next lecture.

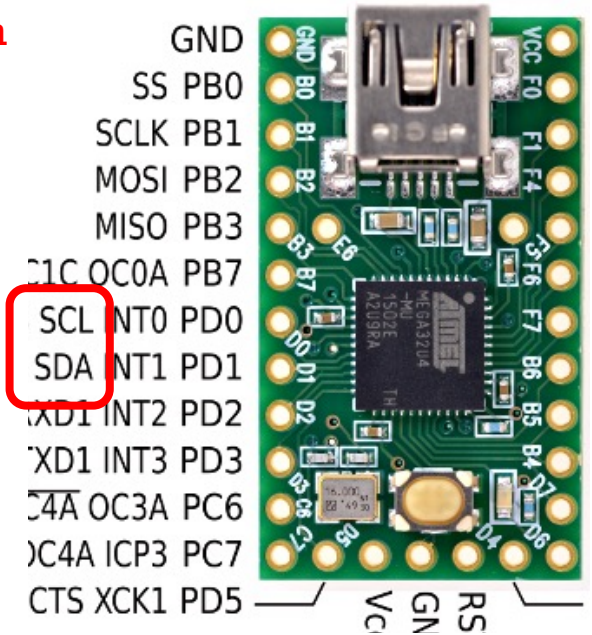
I^2C (Inter-Integrated Circuit)

- Invented by Phillips (NXP)
  - Pronounced "I squared C" sometimes I-two-C
  - Inter-IC implies between chips on a board, not cabled.
- 2 wire (plus ground). Sometimes called Two Wire Interface.
- Synchronous, Half-duplex
- Multidrop – multimaster – 7 Bit address space. (~100 devices)
- Higher level protocol than SPI (packets-switched)
- Typically 40,000 bytes per sec speed (400Kbit/sec)
  - Newer high speed modes – 1Mbit/sec – 3.4Mbit/sec with special hardware drivers

I2C on Arduino – "Wire" Library (not quite done for ESP32)



According to doc: can be set to any GPIO pin (probably not 34+)

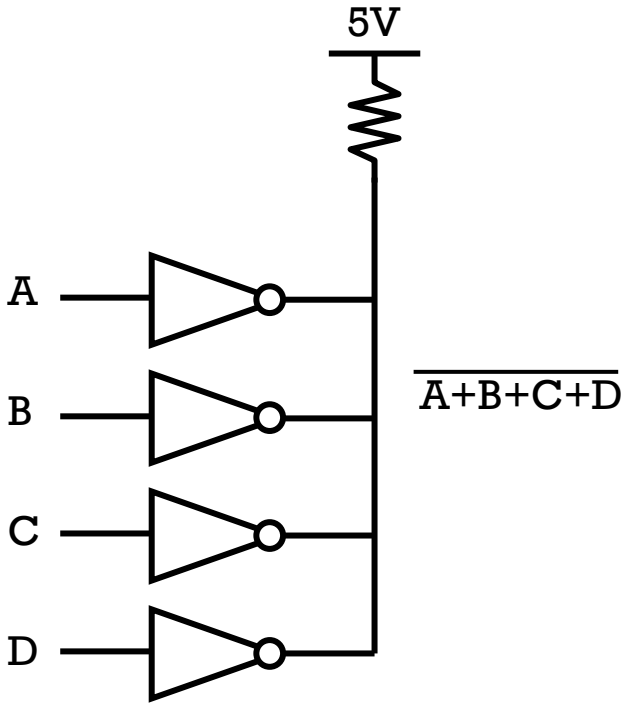


https://www.pjrc.com/teensy/td_libs_Wire.html

Multidrop Wired Bus

- Simplest form of multidrop network
  - "Wired-OR" (low if any is output is low)
  - Can broadcast one output to many inputs
  - All drivers use open collector or open drain
  - What limits # of drops on this network?
- Simplest form of multimaster network
  - No added drivers
  - Very low cost (just add pull up resistor)
  - Collisions are safe
  - I2C uses wired-or

Q7: What is potentially wrong with this circuit?

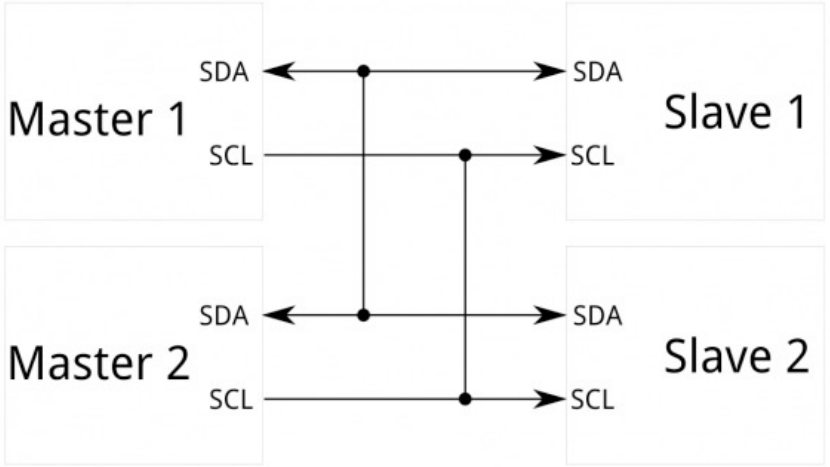


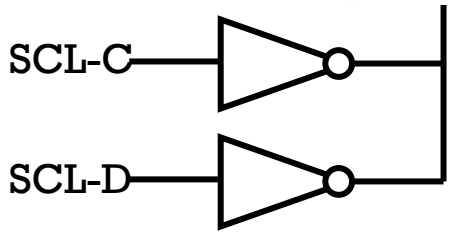
Open Collector/Open Drain output

I2C Open-Drain (wired-OR)

- Two wires (plus ground)
  - SCL (clock line)
  - SDA (data line)

- Master drives Clock line low, but slaves can 'stretch' clock by also pulling SCL line low to slow it down.
- Can easily use 3.3V or 5V as pull–ups can float to any V





Ex: 4 devices driving SCL line

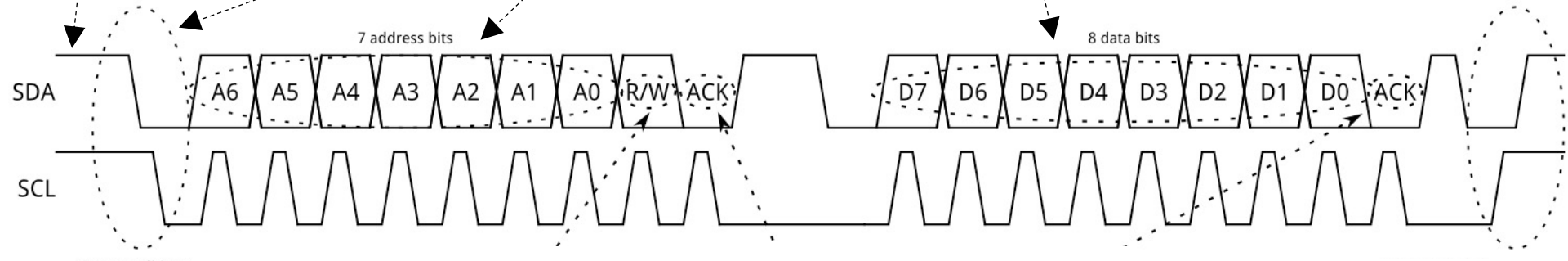
I2C Line Protocol

Nominal state, all lines float high

Master Asserts Start condition

Master drives sequence for address of target slave

Data is transmitted Direction (depending on read/write)

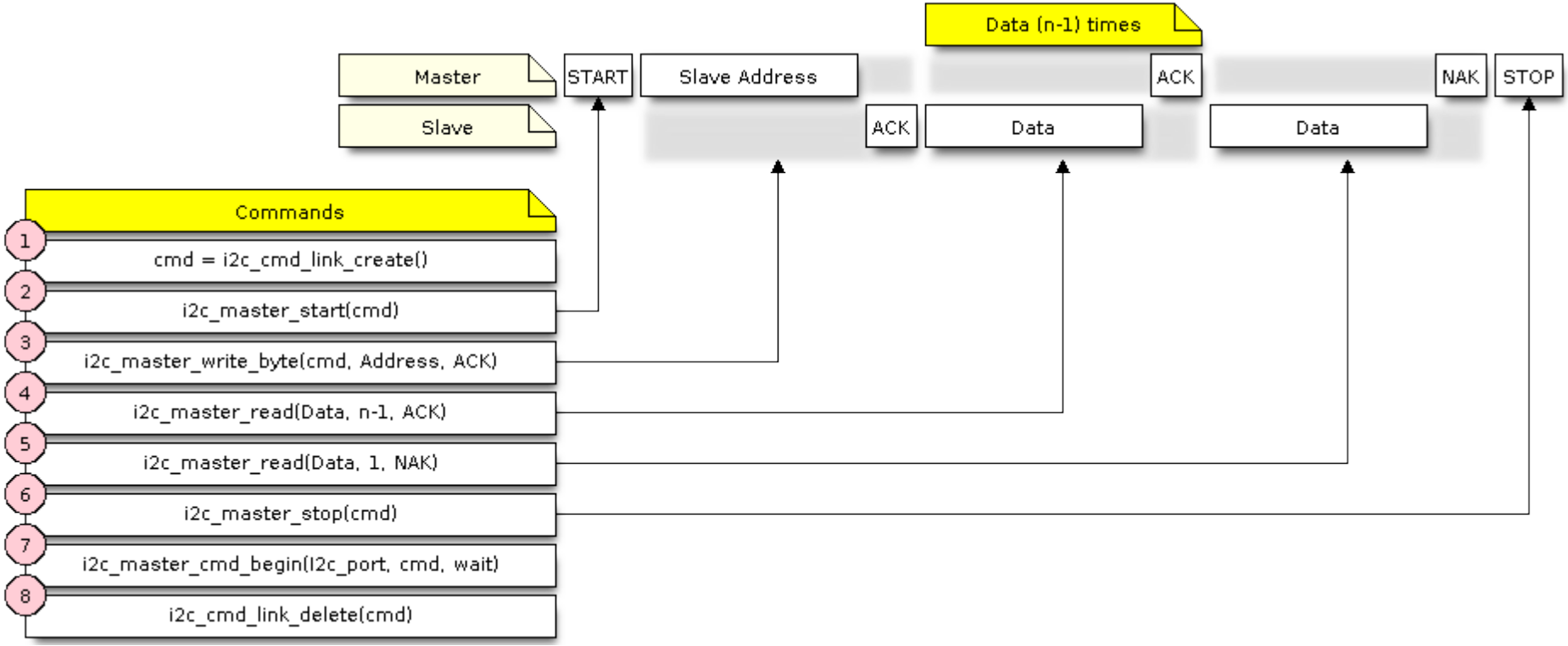


Last bit indicates Write or Read

Slave Acknowledges (or Not)

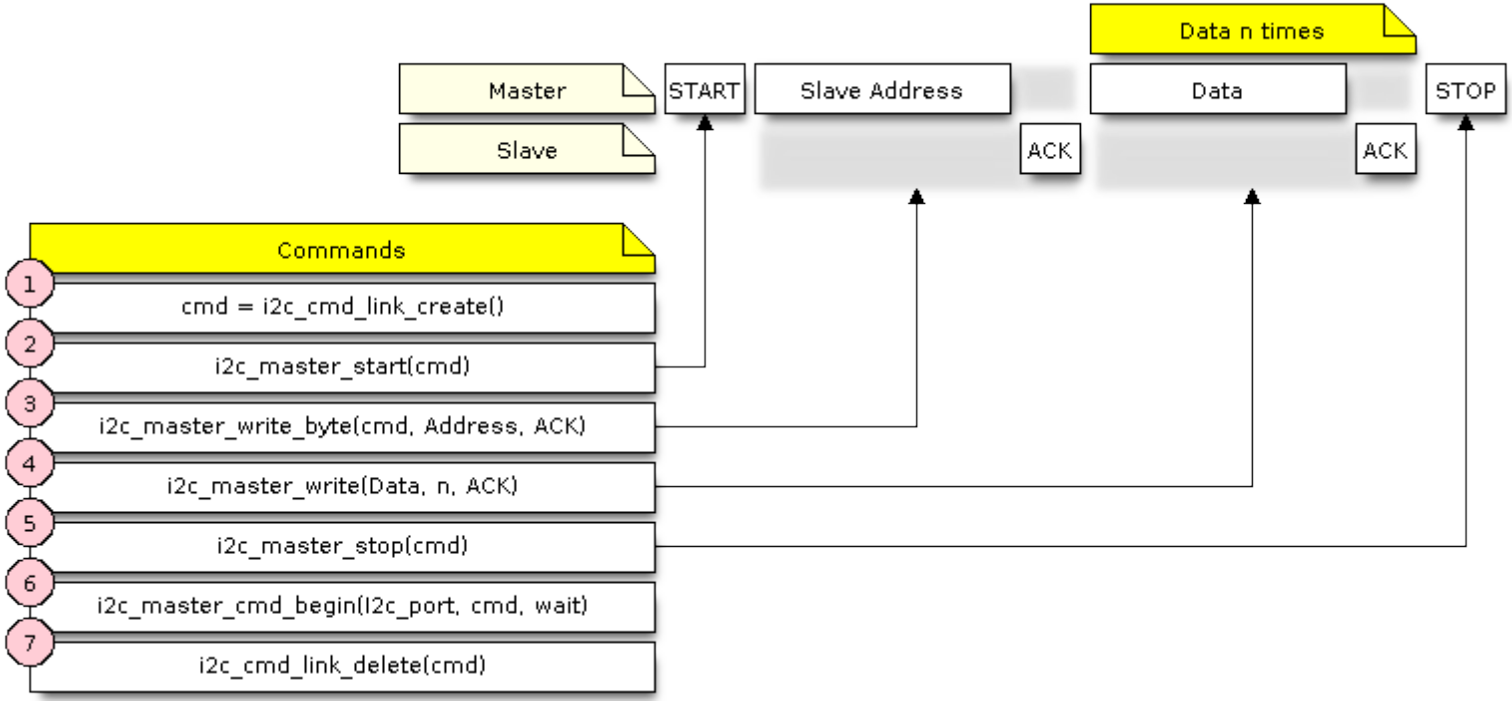
Master Asserts Stop condition

ESP32 Master Read Sequence



https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/peripherals/i2c.html

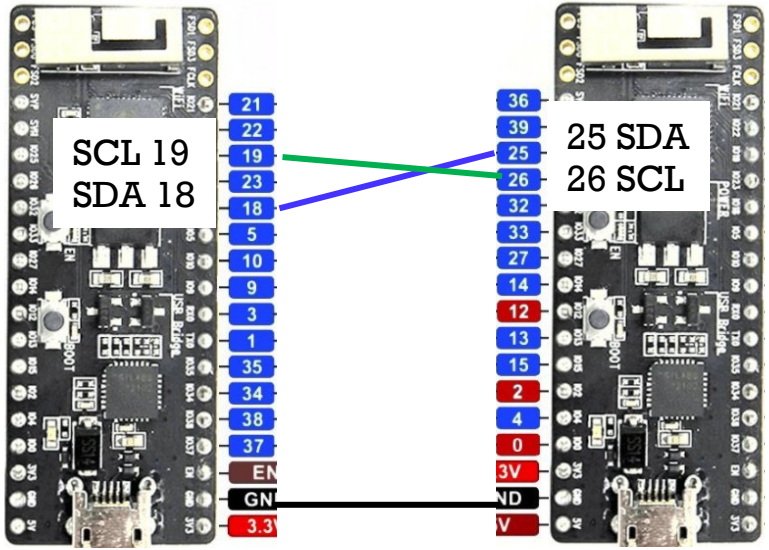
ESP32 Master Write Sequence



https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/peripherals/i2c.html

ESP32 I2C Demo –on canvas [Not Arduino Wire Library]

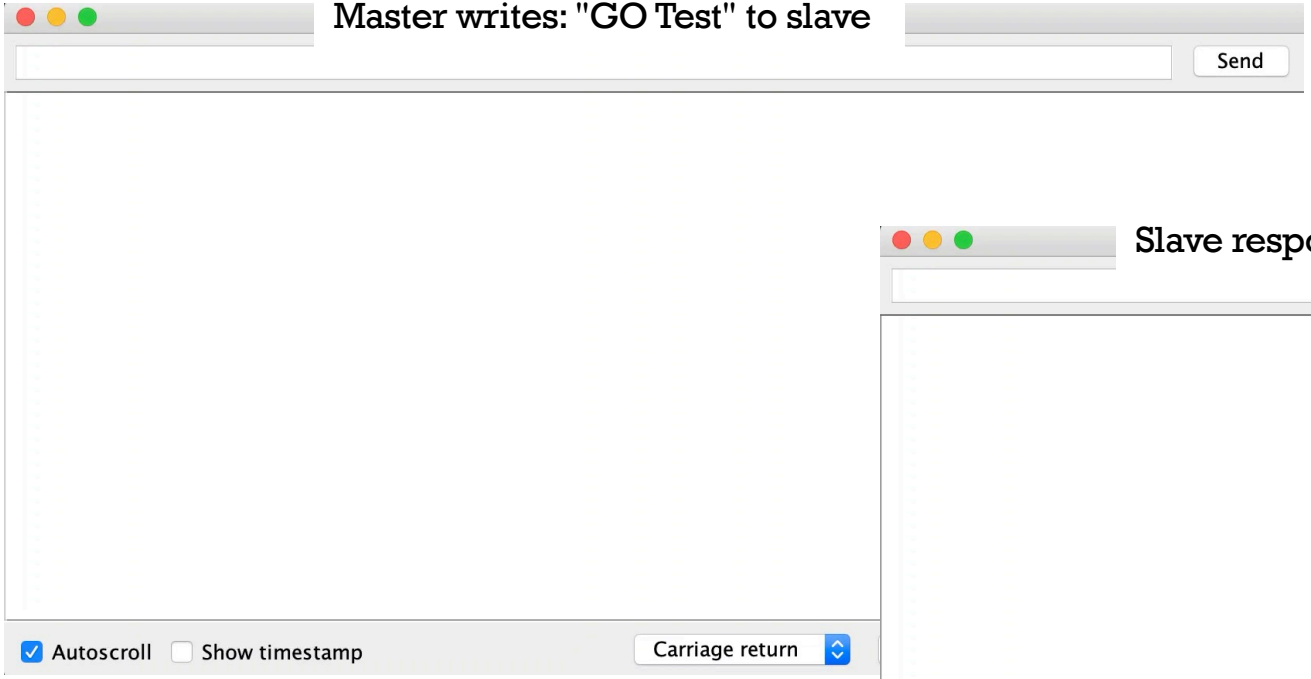
1. Read slave buffer
   Print result
2. Delay 1 sec
3. Write slave
   Print sent data
4. Delay 1 sec
5. Repeat

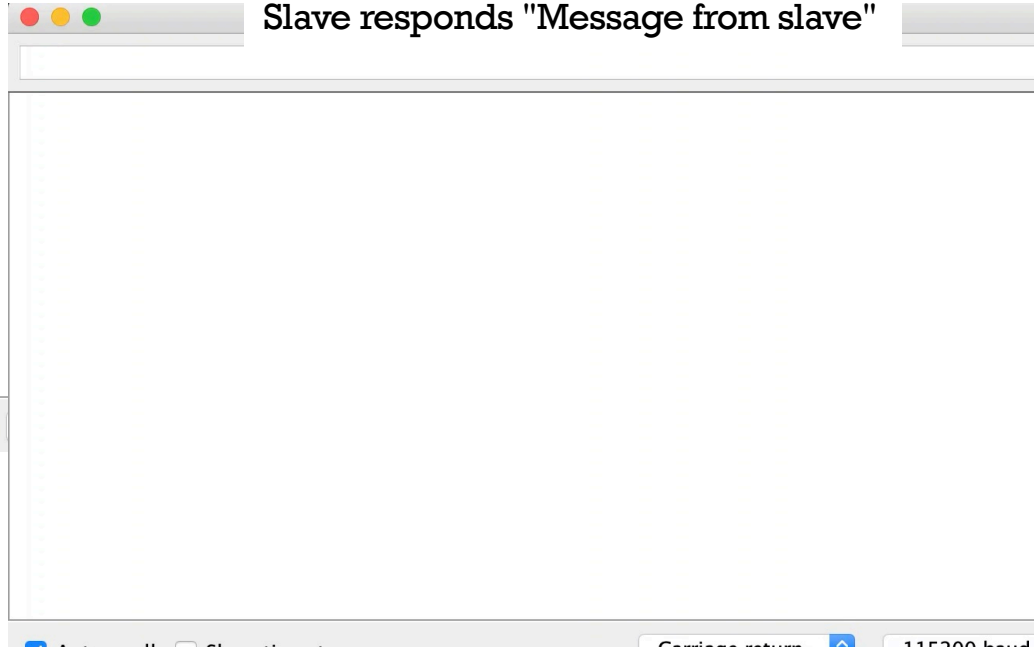


Master Slave

1. Read master
2. If something recv
   1. Print result
   2. Write to master
      Print sent data
3. Repeat

I2C demo (on canvas)





ESP32 IDF (I2C support)

Maximum wait time in ticks

```
i2c_slave_read_buffer(i2c_port_t n, uint8_t* data, size s, TickType_t t)
i2c_slave_write_buffer(i2c_port_t n, uint8_t* data, size s, TickType_t t)

i2c_master_write_slave(i2c_port_t n, uint8_t* data, size s)
i2c_master_read_slave(i2c_port_t n, uint8_t* data, size s)
```

I2C_NUM_0 or I2C_NUM_1

Pointer to Array of Data bytes data[]

integer number of data bytes

Espressif IDF I2C code documentation

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/i2c.html

I2C ESP32 to ESP32 Master

```
... // lots of init stuff before this code
// Canvas->files->resources->ESP32 Arduino Lecture Examples / i2c_master_lecture.ino
#define I2C_MASTER_SCL_IO (gpio_num_t)19
#define I2C_MASTER_SDA_IO (gpio_num_t)18

uint8_t data_wr[] = "GO Test     ";
uint8_t data_rd[DATA_LENGTH];

void setup() {
  Serial.begin(115200);
  i2c_master_init();        // can assign most GPIO pins to be SDA or SCL
}

void loop() {   // check    I2CPort    buffer    amount of bytes to read
  if (i2c_master_read_slave(I2C_NUM_1, data_rd, DATA_LENGTH) == ESP_OK)
    Serial.printf("Read: %s\n",data_rd);
  delay(1000);  // write    I2CPort    buffer    amount of bytes to write
  if (i2c_master_write_slave(I2C_NUM_1, data_wr, DATA_LENGTH) == ESP_OK)
    Serial.printf ("Write: %s\n",data_wr);
  delay(1000);
}
```

I2C ESP32 to ESP32 Slave

```
... // lots of init stuff before this code
// Canvas->files->resources->ESP32 Arduino Lecture Examples / i2c_slave_lecture.ino
#define I2C_SLAVE_SCL_IO (gpio_num_t)26
#define I2C_SLAVE_SDA_IO (gpio_num_t)25

uint8_t data[] = "Message from slave     ";
uint8_t data_rd[DATA_LENGTH];

void setup() {
  Serial.begin(115200);
  i2c_slave_init();         // can assign most GPIO pins to be SDA or SCL
}

void loop() {    // check    I2CPort,   buffer,  bytes to read, time to wait
  if (i2c_slave_read_buffer(I2C_NUM_0, data_rd, RW_TEST_LENGTH, 0) > 0 ) {
    Serial.printf("READ: %s\n",data_rd);
    if (i2c_slave_write_buffer(I2C_NUM_0, data, 10, 10/portTICK_RATE_MS) !=0 )
      Serial.printf("WRITE: %s\n",data);
  } // otherwise no data to read
  delay(10); // pretend we're doing something else here.
}
```

I2C Summary

- I2C is very popular comm protocol.
- Multidrop, multi-master, half-duplex, synchronous serial communication
- Can use two wires (plus ground)
- ESP32 Arduino I2C is buggy, use ESP-IDF routines shown here, or find ESP32 compatible arduino libraries.

Answer in CHAT

Answer how you feel about each topic below with:

1. I don't understand this topic at all
2. I don't know now, but know what to do to get by
3. I understand some, but expect to get the rest later
4. I understand completely already

A. Ranging sensors

B. Master-Slave vs Peer-Peer, Async vs Sync, Half/Full duplex

C. Using I2C on ESP32 if needed.