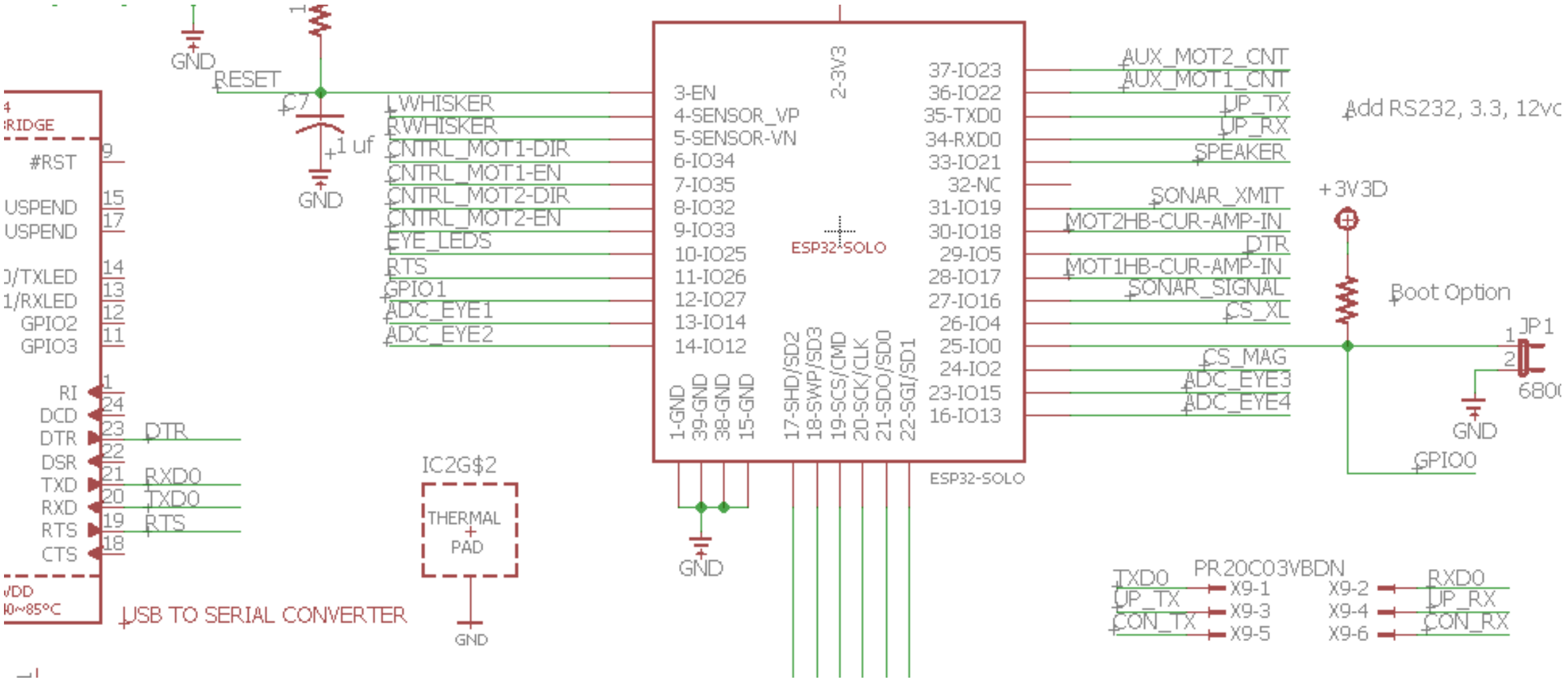
Silicon Valley Forth Interest Group

Dec 2019

Forth, AI for Intelligent Machines

- AIBOT Project
- System Verilog
- FPGA AI Robot Controller
- MadeInSpace

AI BOT Current PCB V1_0a



F32 Bootstrap Configuration

Table 3: Strapping Pins

Voltage of Internal LDO (VDD_SDIO)					
Pin	Default	3.3 V		1.8 V	
MTDI	Pull-down	0		1	
Booting Mode					
Pin	Default	SPI Boot		Download Boot	
GPIO0	Pull-up	1		0	
GPIO2	Pull-down	Don't-care		0	
Enabling/Disabling Debugging Log Print over U0TXD During Booting					
Pin	Default	U0TXD Toggling		U0TXD Silent	
MTDO	Pull-up	1		0	
Timing of SDIO Slave					
Pin	Default	Falling-edge Sampling Falling-edge Output	Falling-edge Sampling Rising-edge Output	Rising-edge Sampling Falling-edge Output	Rising-edge Sampling Rising-edge Output
MTDO	Pull-up	0	0	1	1
GPIO5	Pull-up	0	1	0	1

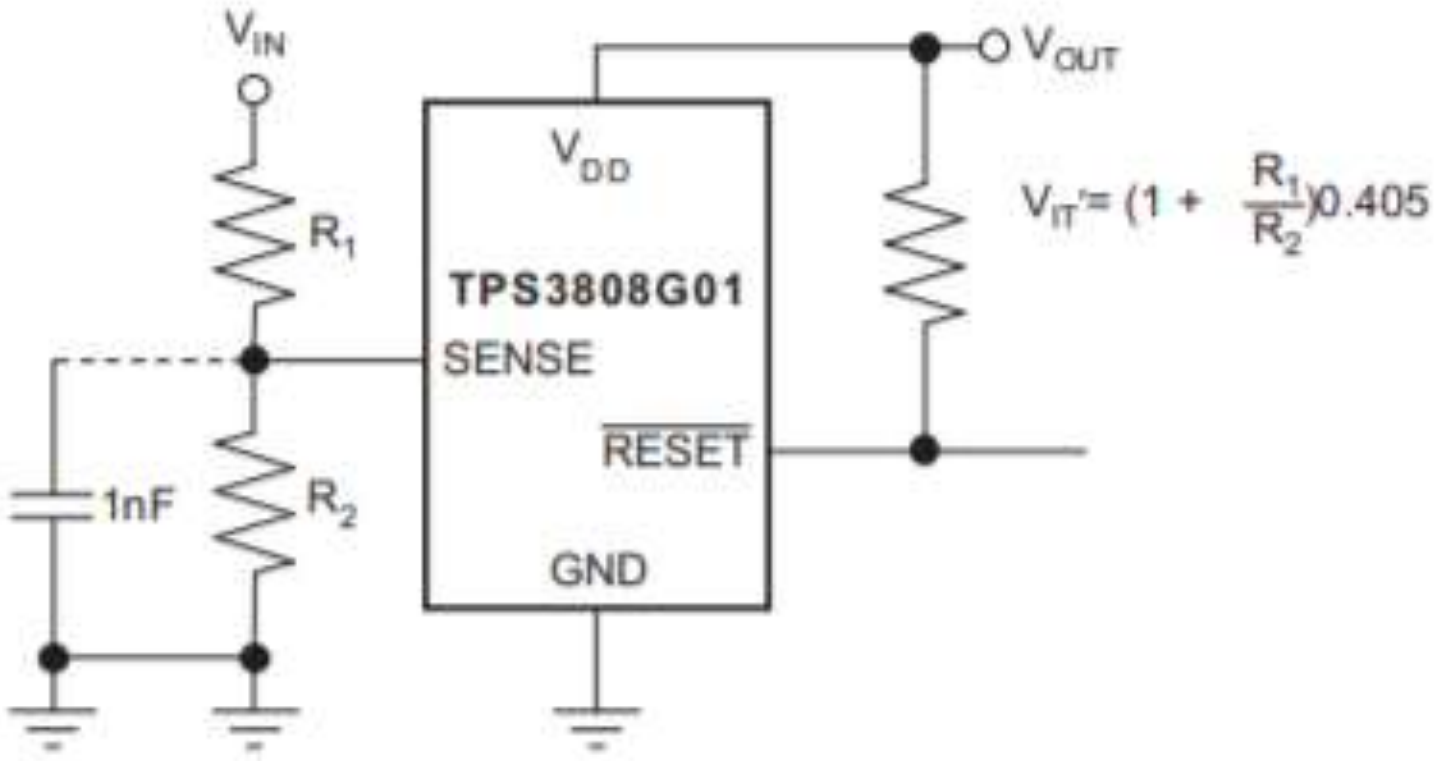
TX Serial Line Harmonic Suppression

2.1.8 UART

Users need to connect a 499 Ω resistor to the U0TXD line in order to suppress the 80 MHz harmonics.

Figure 10: ESP32 UART

Power On Reset



AI BOT Development Plan V1_0b

- Schematic Changes
- PCB Layout Changes
- Spin 4 Boards with only CPU and Serial/Booting circuits
- Load Flash Testing

Best SOM Placement for RF

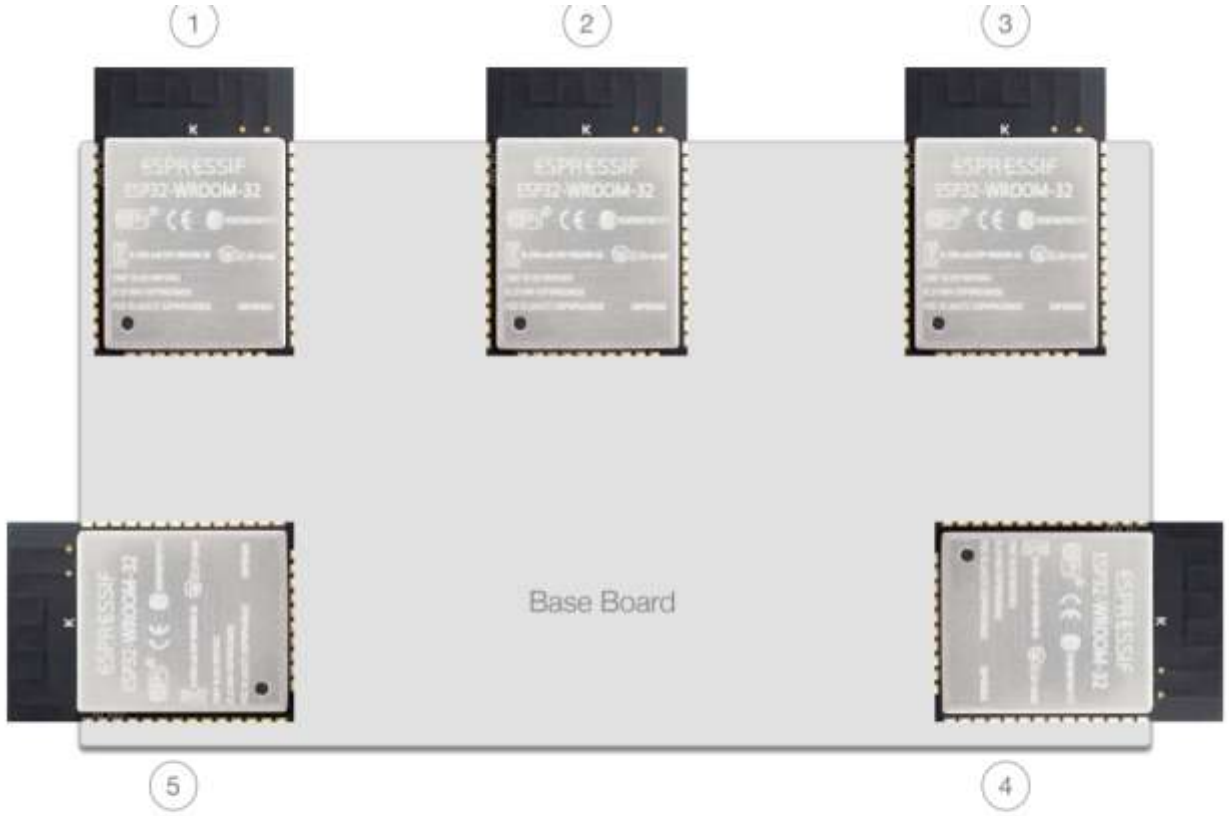


Figure 12: ESP32 Module Antenna Position on Base Board

FPGAs for Embedded Systems

- Field Programmable Gate Arrays
- Easy to Build MASSIVELY Parallel Processors
- Every “Process” is a Tiny Custom Function Processor
- Many Hundreds or Thousands of “Processors” running, SIMULATANEOUSLY!

FPGAs for Embedded Systems

- Field Programmable Gate Arrays
- Easy to Build MASSIVELY Parallel Processors
- Every “Process” is a Tiny Custom Function Processor
- Many Hundreds or Thousands of “Processors” running, SIMULATANEOUSLY up to 250MHZ each

FPGAs for Embedded Systems

- Perfect for Intelligent Robotic Systems
- Robots REQUIRE the Control System to manage dozens of Sensors and Motors: SIMULATANEOUSLY
- Multitasking on CPUs is NOT ENOUGH for Truly Deterministic and Intelligent Response

FPGA Programming

RTL: Register Transfer Logic

VS

Behavioral Modeling

FPGA Programming

- Behavioral Modeling offers much higher Levels of Abstraction
- Easily Build complex Systems of FPGAs
- VHDL was the Preferred Digital Logic Modelling Language since 1990.
- Verilog was the Preferred Language for ASIC Designers

SILICON VALLEY FIG

System Verilog

- System Verilog 2017
- RTL Modeling with System Verilog by Stuart Sutherland
- Based on C Language, Easy for Programmers
- System Verilog is now Preferred Language for new Projects

System Verilog

```
`define DATA_BUS_WIDTH 31
`define ADDR_BUS_WIDTH 31
`define CTR_WIDTH 26
`define SLOT6_ADDR `ADDR_BUS_WIDTH'h70006000
```

System Verilog

```
module APB_outReg (
    input logic PCLK,
    input logic PSEL,
        input logic PENABLE,
        input logic PWRITE,
        input logic [`ADDR_BUS_WIDTH:0] PADDR,
    input logic [`DATA_BUS_WIDTH:0]PWDATAS_4,
        output logic [`DATA_BUS_WIDTH:0] PRDATAS6);
```

System Verilog

```
logic [`DATA_BUS_WIDTH:0] PWDATA_n;
logic PSEL_n, PENABLE_n, PWRITE_n, PCLK_n;
logic [`CTR_WIDTH:0] Counter;
logic [1:0] Q_reg;
logic [7:0] Mode_n;
```

System Verilog

```
assign PSLVERRS6 = 1'b0;
assign PREADYS6 = 1'b1;

assign PWDATA = PWDATA_n;
assign PSEL_n = PSEL;
assign PENABLE_n = PENABLE;
assign PWRITE_n = PWRITE;
assign Q = Q_reg;
assign Mode_n = PWDATAS_4;
```

System Verilog

```
always_ff @(posedge PCLK)
begin
    Counter <= Counter + 1;
if (Counter == 0)
    Q_reg = ~Q_reg;
end
```

System Verilog

```
enum logic [`ARM_ERROR_WIDTH]{
        ARM_POS_OK = 3'b00,
        ARM_POS_ERR = 3'b01,
        ARM_HOME_ERR = 3'b10,
        ARM_OVER_CUR_ERR = 3'b11} ARM_ERRORS;
```

System Verilog

```
typedef struct {

    logic [7:0] Sub_Type;
    logic [`JOINT_DATA_SIZE] Data;

}Arm_Command_Data_t;

Arm_Command_Data_t   Arm_Command_Packet[`NUM_JOINTS_SIZE];
```

System Verilog

```
function automatic logic [`ARM_ERROR_WIDTH] Process_Velocity();
begin

    Load_Velocity();//Load torque values from RISC V

    for (int i=0;  i<`NUM_OF_JOINTS;  i++)
    begin
        Arm_Command_Packet[i].Data = Process_PID(Joint_PID[i]);
        Arm_Command_Packet[i].Data = Arm_Command_Packet[i].Data + Offset_Torque[i];
    end
    return ARM_POS_OK;

end
endfunction
```

FPGA Forth Robotic Controller

- Use Dr. Ting's FPGA Forth Engine
- ADD: SRAM + Flash (or MRAM)
- Robot Driver Circuits

FPGA Forth AI Robotic

- Add: Truine OS
- Develop New AI Language