

Mithril - FMCW Radar

by

Tomas Esson, Ajay Thakkar, Juan Jimenez

tesson@stevens.edu, athakka5@stevens.edu, jjimene6@stevens.edu

May 9, 2024

© Tomas Esson, Ajay Thakkar, Juan Jimenez
tesson@stevens.edu, athakka5@stevens.edu, jjimene6@stevens.edu
ALL RIGHTS RESERVED

Table of Contents

1	Introduction	1
2	Project Description	2
2.1	FMCW Radar PCB	3
3	Resources	4
4	Radar Theory	5
5	Part Selection	6
6	PCB	7
6.1	Schematic	7
6.1.1	ORCad Capture Overview	7
6.1.2	Circuitry Good Practices	9
6.1.3	Transmitter	9
6.1.4	Receiver	9
6.2	Layout	9
7	Digital Processing	10
8	Networking	11
9	Results	12
10	Issues	13

List of Tables

List of Figures

2.1	Flowchart of the Mithril system	2
6.1	Component Database Search	7
6.2	Schematic/Footprint Examples	8
6.3	Editing Schematic Parts	8
6.4	Wires and Net Alias	8
6.5	Off Page Connectors	9

Chapter 1

Introduction

The following includes small biographies on all the authors as well as their research interests and projects.

Authors' Biographies

Ajay Thakkar

Ajay Thakkar Ajay Thakkar is a junior majoring in Computer Engineering. He is interested in signal processing and lower level coding. Below you can find his GitHub: <https://github.com/athakkar2>.

Tomas Esson

Tomas Esson is an aspiring Computer Engineering at Stevens Institute of technology. He is an avid surfer and enjoys elegant math proofs. Currently pursuing interests in computer chip design, digital systems implementation, mathematical optimization of computer chips, and electrical engineering.

Juan Jimenez

Juan Jimenez is a Junior Computer Engineering student at the Stevens Institute of technology. Interested in the intersection between Artificial Intelligence, embedded electronics, and software engineering. To see more projects visit the following GitHub link: <https://github.com/jjimene1>

Chapter 2

Project Description

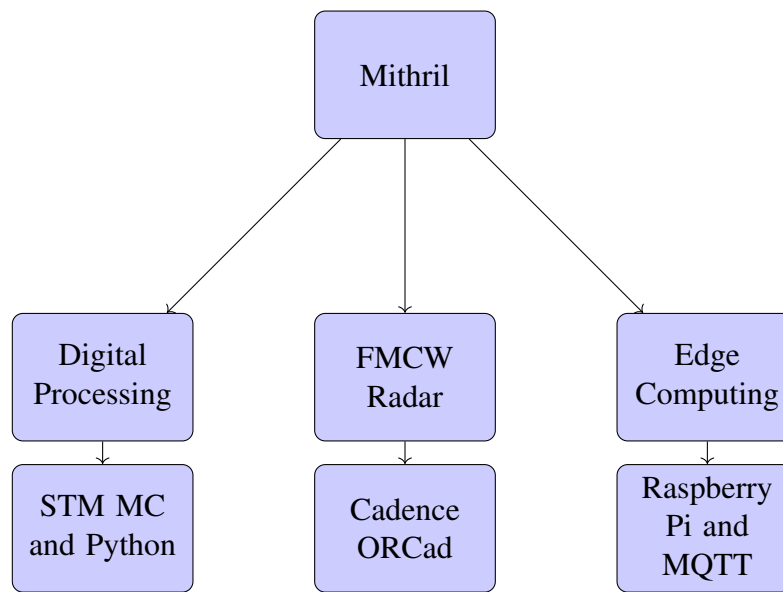


Figure 2.1: Flowchart of the Mithril system

Mithril is a nodal FMCW radar system that incorporates traditional FMCW radar, digital processing, edge computing, and distributed networking. The initial application of this project was for a military context. State of the art missile defense systems cost too much and are too big to be viable in many places, and while our system doesn't compare to systems like the Iron Dome or Patriot System it could allow for at least an alert for civilians to leave possible shelling and bombing sites. The system would include small radar nodes capable of detecting missiles and electronically steering their line of sight via phased array antennas. Multiple nodes could work in tandem to communicate their findings, allowing for a wider range of sight and active tracking of projectiles. The network would be able to function regardless of how many nodes there were, and still continue to function if nodes are destroyed.

As can be seen in Figure 2.1, the radar was designed as a standalone PCB in ORCad, digital processing was handled by STM microcontrollers, and distributed networking is done via Raspberry

Pi's and the MQTT protocol. All of these components were designed, engineered, and interfaced from scratch with a limited budget of 2000 dollars.

2.1 FMCW Radar PCB

The heart of the project is a standalone PCB capable of FMCW radar.

Chapter 3

Resources

Chapter 4

Radar Theory

Chapter 5

Part Selection

Chapter 6

PCB

6.1 Schematic

6.1.1 ORCad Capture Overview

ORCad comes with a schematic software called Capture CIS which we used to create our schematics. As a reference to both us and others, this section is dedicated to providing a small walkthrough to using the software and some nice-to-know features.

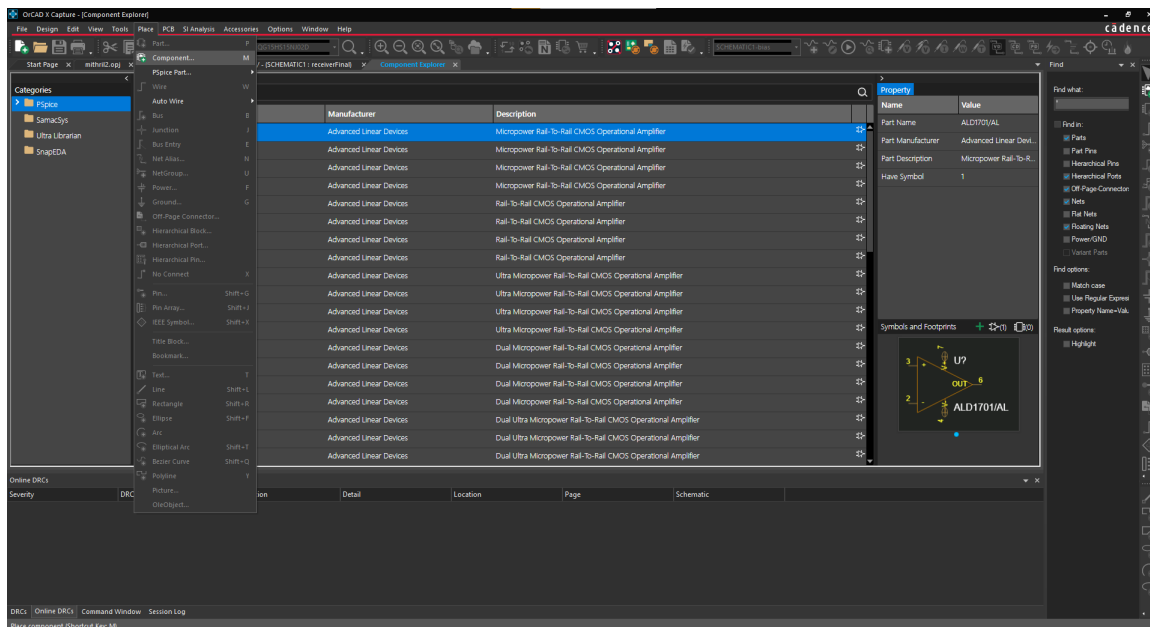


Figure 6.1: Component Database Search

The first important feature we found was the component database, which can be seen in Figure 6.1. By clicking on the icon that has a small chip and cloud will take you to this page which contains subdirectories "PSpice" "SamacSys" "UltraLibrarian" and "SnapEDA". PSpice contains parts that have simulation properties, but we ignored this as we did not know how to use PSpice.

The other three are different databases containing schematic symbols and layout footprints for parts that can be found on major retailers like DigiKey, Mouser, and Arrow.





...	SI8645BB-B-IS	Silicon Labs	SI8645BB-B-IS Silicon Labs, PCB SMT, 4-Channel Digital Isolator 150Mbps, 2.5 kV, 16-Pin SOIC	
...	TMP125AIDBVR	Texas Instruments	±2°C 2.7V to 5.5V digital temperature sensor with SPI interface and -40 to 125°C o	  

Figure 6.2: Schematic/Footprint Examples

You can search for parts in the search fields for the different databases to try to find a part that has a schematic symbol and footprint available. You can see in Figure 6.2 that parts with the symbol and footprint available will have an amp and chip symbol next to them. The box means there is a 3D CAD model associated with them too. If you cannot find the symbol and footprint for a part, we recommend going on Mouser and finding the part, then requesting the symbol and footprint. Usually, the schematic and footprint will be added to SamacSys after a couple of days.

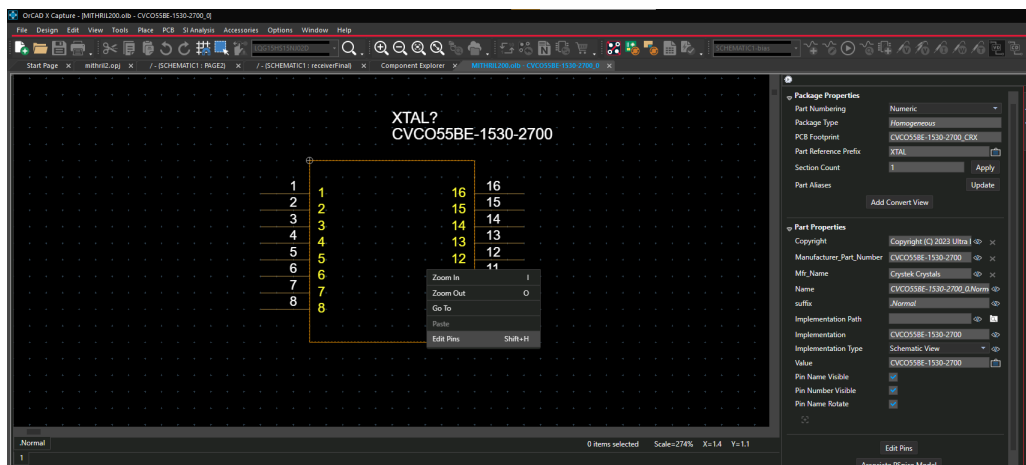


Figure 6.3: Editing Schematic Parts

Sometimes, these symbols and footprints are not correct. If the schematic symbol is incorrectly labeled, you can edit the part, then click edit pins to make sure all the pins are correctly labeled and numbered as can be seen in Figure 6.3.

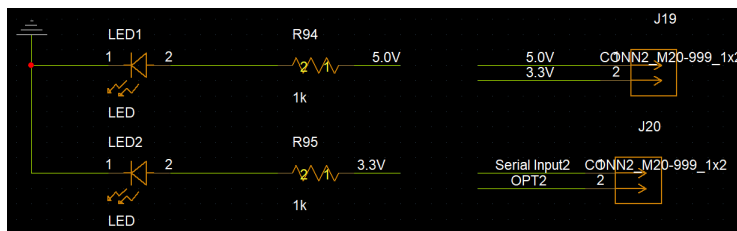


Figure 6.4: Wires and Net Alias

Once you've placed your parts down in the schematic, now comes time to connect everything together. To navigate through the schematic interface, you can use CTRL+Scroll Wheel to zoom in and out, and middle mouse button to scroll left to right.

You can use "w" to enter wiring mode, which lets you place down wires according to your grid size. After wiring, be sure to use "n" to enter net alias mode and assign aliases to your wires. As can be seen in Figure 6.4, there are two non-connected wires both with the alias "5.0V". This effectively connects them since they are under the same alias and will also label them in the layout when routing. Using the net alias helps with things like power where connecting everything that needs power with a wire to your power source would make the schematic a mess. In short, all wires with the same net alias are considered connected.

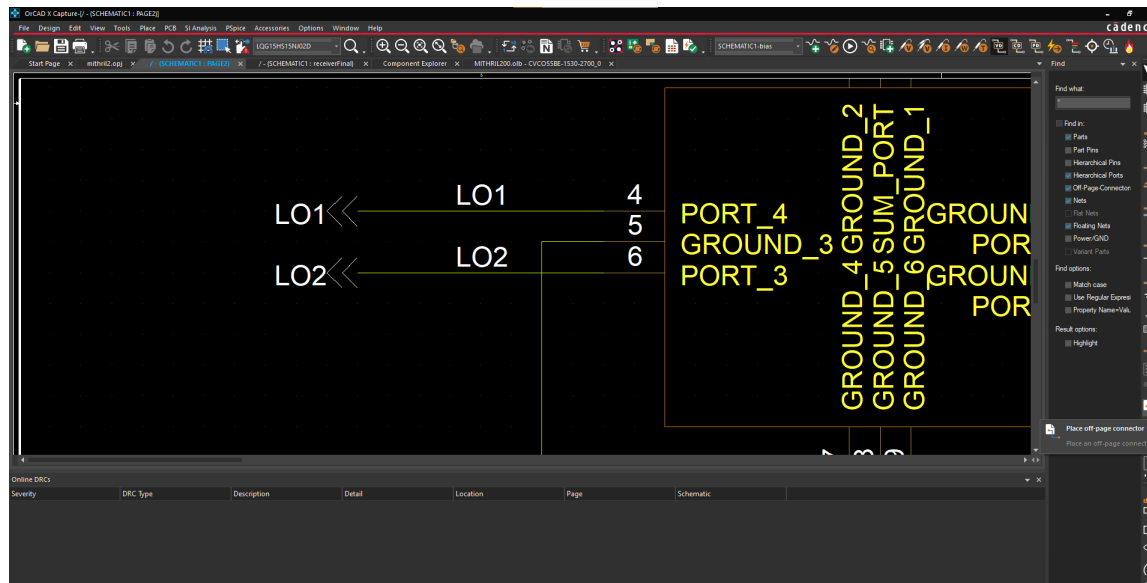


Figure 6.5: Off Page Connectors

If you run out of room or want to separate your schematics into different pages, you can connect wires from different schematic pages using off-page connectors as shown in Figure 6.5. Connect the off-page connector to a wire and name it the same on both schematic pages to have the wires connect.

6.1.2 Circuitry Good Practices

6.1.3 Transmitter

6.1.4 Receiver

6.2 Layout

Chapter 7

Digital Processing

Chapter 8

Networking

Chapter 9

Results

Chapter 10

Issues

Index

Chapter

Digital Processing, 10

Introduction, 1

Issues, 13

Networking, 11

Part Selection, 6

PCB, 7

Project Description, 2

Radar Theory, 5

Resources, 4

Results, 12

Digital Processing, 10

Introduction, 1

Issues, 13

Networking, 11

Part Selection, 6

PCB, 7

Project Description, 2

Radar Theory, 5

Resources, 4

Results, 12