DAFTAR PUSTAKA

Ashraf, Nausheen, Shahzad Amin Sheikh, Sajjad Ahmad Khan, Ibraheem Shayea, and Marium Jalal. 2021. "Simultaneous Wireless Information and Power Transfer with Cooperative Relaying for Next-Generation Wireless Network: a Review." *IEEE.*

Balanis, A. C. 2016. *ANTENNA THEORY: ANALYSIS AND DESIGN (4th ed.).* Hoboken: John Wiley & Sons.

Bi, Suzhi, Chin Keong Ho, and Rui Zhang. 2015. "Wireless Powered Communication" Opportunities and Challenges." *IEEE.*

Dwiwulandari, Muthia, Elyas Palantei, Intan Sari Areni, and Zulfahmi Rizal. 2024. "A 2.4 GHz Wireless Power Transmission: Design and Testing." *IEEE.*

2021. "ESP 32 Series Datasheet." https://www.espressif.com/en/support/download/documents.

Garcia, Joan J. 2022. "Considerations for the Design and Implementation of Ambient." *Applied Science* 1-11.

Huang, Jun, Cong-Cong Xing, and Chonggang Wang. 2017. "Simultaneous Wireless Information and Power Transfer: Technologies, APplications, and Research Challenges." *IEEE.*

Huang, Y., and K. Boyle. 2008. *Antennas From Theory to Practice.* Singapore: John Wiley & Sons.

Karami, M. A., and Kambis Moez. 2019. "Systematic Co-Design of Matching Networks." *IEEE Transactions Circuit Systems* 3238-3251.

Kwan, Derrick Wing, Trung Duong, Caijun Zhong, and Robert Schober. 2019. *Wireless Information and Power Transfer: Theory and Practice.* New Jersey: Wiley.

Lu, X., P. Wang, and D. Niyato. 2015. "Wireless Networks with RF Energy Harvesting: A Contemporary Survey." *IEEE.*

Milligan, Thomas A. 2005. *Modern Antenna Design (Second Edition).* New Jersey: John Wiley & Sons.

Nizam, Muhammad, Haris Yuana, and Wulansari Zunita. 2022. "Mikrokontroler ESP32 sebagai Alat Monitoring Pintu Berbasis Web." *JATI.*

Palantei, Elyas, Zulfahmi Rizal, Intan Sari Areni, Muthia Dwiwulandari, Regita Pramestia, and Josaphat Tetuko Sri Sumantyo. 2024. "An Efficient Power Transfer Technique of SIMO Antennas Network." *IEEE.*

Perera, Tharindu Dilshan, Dush Nalin Jayakody, and Shree Khrisna Sharma. 2017. "Simultaneous Wireless Information and Power Transfer (SWIPT): Recent Advantages and Future Challenges." *IEEE.*

Piovesan, Nicola, Angel Fernandez Gambin, Marco Miozzo, Michelle Rossi, and Paolo Dini. 2018. "Energy Sustainable Paradigms and Methods for Future Mobile Network: a Survey." *IEEE*.

Said, M. N., Z. Zakaria, M. N. Husain, M. M. Yunus, M. Abu, and M. H. Husain. 2014. "A Review of Rectifier Designs for RF Energy Harvesting System." *Australian Journal of Basic and Applied Science* 180-188.

Shannon, Claude. 1948. "A Mathematical Theory of Communication." *The Bell System Technical Journal.*

Shannon, Claude. 1998. "Communication in the Presence of Noise." *IEEE*.

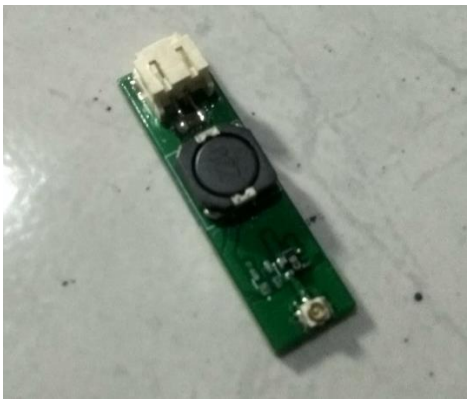
Skolnik, Merrill I. 2008. *RADAR HANDBOOK*. New York: The McGraw-Hill Companies.

Vyas, Sandeep Kumar, Rajendra Kumar Verma, Shivendra Maurya, and Vindhyavansini Prasad Singh. 2015. "Review of Magnetron Developments." *Frequenz* 455-462.

Xu, Ziyue, Adam Khalifa, Ankit Mittal, Mehdi Nasollahpourmotlaghzanjani, Etiene Cummings Ralph, Nian Xiang Sun, Aatmesh Shrivastava, and Sydney S Cash. 2022. "Analysis and Design Methodology of RF Energy Harvesting." *IEEE Open J Circuit Sys* 82-96.

LAMPIRAN

Lampiran 1 Hasil Fabrikasi Perangkat



Lampiran 2 *Source Code* GUI APP

```
      import tkinter as tk
import serial
import serial.tools.list_ports
import threading
import time
from tkinter import filedialog
import openpyxl

ser = None  # Global variable to hold the serial connection
connected = False  # Global variable to track connection status

data_dict = {"Current": [], "Input Voltage": [], "Besar
Perpindahan": []}  # Dictionary to store received data

def connect_serial():
    global ser, connected
    port = port_var.get()
    baudrate = baudrate_var.get()
    try:
        ser = serial.Serial(port, int(baudrate))
        connected = True
        update_connect_button()
        print(f"Connected to {port} at {baudrate} baud")
    except Exception as e:
        print(f"Error: {e}")

def disconnect_serial():
    global ser, connected
    if ser:
        ser.close()
        ser = None
    connected = False
    update_connect_button()
    print("Disconnected")

def update_connect_button():
    if connected:
        connect_button.config(text="Disconnect",
command=disconnect_serial)
    else:
        connect_button.config(text="Connect",
command=connect_serial)

def update_ports():
```

```
    port_list = [port.device for port in
serial.tools.list_ports.comports()]
    port_var.set(port_list[0])  # Set the default port
    port_menu['menu'].delete(0, 'end')  # Clear existing menu items
    for port in port_list:
        port_menu['menu'].add_command(label=port, command=lambda
p=port: port_var.set(p))

def refresh_ports():
    port_menu['menu'].delete(0, 'end')  # Clear existing menu items
    update_ports()

def update_baudrates():
    baudrate_list = ['115200', '9600']  # Add more baud rates if
needed
    baudrate_var.set(baudrate_list[0])  # Set the default baud rate
    baudrate_menu['menu'].delete(0, 'end')  # Clear existing menu
items
    for baudrate in baudrate_list:
        baudrate_menu['menu'].add_command(label=baudrate,
command=lambda b=baudrate: baudrate_var.set(b))

def update_data():
    while True:
        if ser:
            data = ser.readline().decode('utf-8').strip()
            print("Received data:", data)  # Debug line
            for key in data_dict:
                if data.startswith(key):
                    value = float(data.split(":")[1])
                    data_dict[key].append(value)
                    app.after(0, update_labels)  # Update labels on
the main thread
                    break
        time.sleep(0.1)  # Add a short delay

def update_labels():
    current_label.config(text=f"Current: {data_dict['Current'][-1]
if data_dict['Current'] else 'N/A'}")
    voltage_label.config(text=f"Voltage: {data_dict['Input
Voltage'][-1] if data_dict['Input Voltage'] else 'N/A'}")
    displacement_label.config(text=f"Displacement: {data_dict['Besar
Perpindahan'][-1] if data_dict['Besar Perpindahan'] else 'N/A'}")
    target_displacement_label.config(text=f"Target Displacement:
{target_displacement.get()} mm")

def move_stepper():
```

```
    target_mm = target_displacement.get()
    # Replace this value with the actual pitch in millimeters per
revolution
    pitch_mm_per_revolution = 40.0  # Example value, replace with
your motor's pitch
    target_steps = int(target_mm / pitch_mm_per_revolution * 200)
    if ser:
        ser.write(f"MOVE:{target_steps}\n".encode('utf-8'))
        print(f"Moving stepper to {target_steps} steps")

def save_data():
    file_path =
filedialog.asksaveasfilename(defaultextension=".xlsx",
filetypes=[("Excel Files", "*.xlsx")])
    if file_path:
        workbook = openpyxl.Workbook()
        sheet = workbook.active

        headers = list(data_dict.keys())
        sheet.append(headers)

        data_rows = zip(*[data_dict[key] for key in headers])
        for row in data_rows:
            sheet.append(row)

        workbook.save(file_path)
        print(f"Data saved to {file_path}")

app = tk.Tk()
app.title("MWPT Measurement APP")

port_var = tk.StringVar()
port_menu = tk.OptionMenu(app, port_var, "")
port_menu.grid(row=0, column=1)
update_ports()

refresh_button = tk.Button(app, text="Refresh Serial Port",
command=refresh_ports)
refresh_button.grid(row=0, column=2)

baudrate_var = tk.StringVar()
baudrate_menu = tk.OptionMenu(app, baudrate_var, "")
baudrate_menu.grid(row=0, column=3)
update_baudrates()

connect_button = tk.Button(app, text="Connect",
command=connect_serial)
```

```
connect_button.grid(row=0, column=0)
update_connect_button()  # Initialize the button label and behavior

current_label = tk.Label(app, text="Current: N/A")
current_label.grid(row=1, column=0, columnspan=4)

voltage_label = tk.Label(app, text="Voltage: N/A")
voltage_label.grid(row=2, column=0, columnspan=4)

displacement_label = tk.Label(app, text="Displacement: N/A")
displacement_label.grid(row=3, column=0, columnspan=4)

target_displacement = tk.DoubleVar(value=0.0)  # Variable to store
target displacement value
target_displacement_label = tk.Label(app, text="Target Displacement
(mm):")
target_displacement_label.grid(row=4, column=0, columnspan=2)

target_displacement_entry = tk.Entry(app,
textvariable=target_displacement)
target_displacement_entry.grid(row=4, column=2)

move_button = tk.Button(app, text="Move Stepper",
command=move_stepper)
move_button.grid(row=4, column=3)

save_button = tk.Button(app, text="Save Data", command=save_data)
save_button.grid(row=5, column=0, columnspan=4)

update_thread = threading.Thread(target=update_data)
update_thread.daemon = True
update_thread.start()

app.mainloop()
```