LAMPIRAN B

PROGRAM

Sensor Anemometer

```
#define jml_celah 18
#define period 1000
#define delaytime 1000
#define radio 90
const float pi = 3.14159265;
const unsigned long jeda = 3000;
unsigned int Sample = 0;
unsigned int counter = 0;
unsigned int RPM = 0;
unsigned long awalPrint = 0;
const int averageValue = 500;
long int sensorValue = 0;  // variable to store the sensor value read
ICACHE_RAM_ATTR void addcount() {
  counter++;
}
void main_anemo() {
    Sample++;
    wind_velocity();
    RPMcalc();
    WindSpeed();
    Serial.print("Wind speed: ");
    Serial.print(speedwind);
    Serial.print("  [m / s]");
    Serial.println();
}
// Measure wind speed
void wind_velocity() {
  speedwind = 0;
  counter = 0;
  attachInterrupt(digitalPinToInterrupt(windPin), addcount, CHANGE);
  unsigned long millis();
  long startTime = millis();
  while (millis() < startTime + period) {}
  detachInterrupt(digitalPinToInterrupt(windPin));
}
void RPMcalc() {
  RPM = ((counter / jml_celah) * 60) / (period / 1000);
}
```

```
}

void WindSpeed() {
  speedwind = ((2 * pi * radio * RPM) / 60) / 1000;
}
```

Sensor Pzem

```
void preTransmission()
{
  if (millis() - startMillis1 > 5000)  // Wait for 5 seconds as ESP Serial cause start up code crash
  {
    digitalWrite(MAX485_RE, 1);
    digitalWrite(MAX485_DE, 1);
    delay(1);
  }
}
void postTransmission()
{
  if (millis() - startMillis1 > 5000)
  {
    delay(3);
    digitalWrite(MAX485_RE, 0);
    digitalWrite(MAX485_DE, 0);
  }
```

```
}
void setShunt(uint8_t slaveAddr)
{
  static uint8_t SlaveParameter = 0x06;
  static uint16_t registerAddress = 0x0003;
  uint16_t u16CRC = 0xFFFF;
  u16CRC = crc16_update(u16CRC, slaveAddr);
  u16CRC = crc16_update(u16CRC, SlaveParameter);
  u16CRC = crc16_update(u16CRC, highByte(registerAddress));
  u16CRC = crc16_update(u16CRC, lowByte(registerAddress));
  u16CRC = crc16_update(u16CRC, highByte(NewshuntAddr));
  u16CRC = crc16_update(u16CRC, lowByte(NewshuntAddr));
  preTransmission();
  PZEMSerial.write(slaveAddr);
  PZEMSerial.write(SlaveParameter);
  PZEMSerial.write(highByte(registerAddress));
  PZEMSerial.write(lowByte(registerAddress));
  PZEMSerial.write(highByte(NewshuntAddr));
  PZEMSerial.write(lowByte(NewshuntAddr));
  PZEMSerial.write(lowByte(u16CRC));
  PZEMSerial.write(highByte(u16CRC));
  delay(10);
  postTransmission();
  delay(100);
}
void changeAddress(uint8_t OldslaveAddr, uint8_t NewslaveAddr)
{
  static uint8_t SlaveParameter = 0x06;
  static uint16_t registerAddress = 0x0002;
  uint16_t u16CRC = 0xFFFF;
  u16CRC = crc16_update(u16CRC, OldslaveAddr);
  u16CRC = crc16_update(u16CRC, SlaveParameter);
  u16CRC = crc16_update(u16CRC, highByte(registerAddress));
  u16CRC = crc16_update(u16CRC, lowByte(registerAddress));
  u16CRC = crc16_update(u16CRC, highByte(NewslaveAddr));
  u16CRC = crc16_update(u16CRC, lowByte(NewslaveAddr));
  preTransmission();
  PZEMSerial.write(OldslaveAddr);
  PZEMSerial.write(SlaveParameter);
  PZEMSerial.write(highByte(registerAddress));
  PZEMSerial.write(lowByte(registerAddress));
  PZEMSerial.write(highByte(NewslaveAddr));
  PZEMSerial.write(lowByte(NewslaveAddr));
  PZEMSerial.write(lowByte(u16CRC));
  PZEMSerial.write(highByte(u16CRC));
  delay(10);
  postTransmission();
  delay(100);
}
```