מה היא הבחירה הטובה ביותר בעת בניית מכונות מודלריות

Yves De La Broise, KINGSTAR Software Architect

ממשק המכונה

מאחר שמכונות פועלות לבד לעתים נדירות, יש צורך לשלב אותן עם מכונות אחרות, ועם מערכת SCADA ומערכת MES. ההחלטה לגבי תבניות fieldbus ונתונים שמשמשים להחלפת מידע עם מכונות אחרות ועם המפעל, מתבצעת על ידי הלקוח הסופי (ולא על ידי בונה המכונה). המשמעות היא שהממשק שנמצא בתוך המכונה ומשמש את הלקוח, חייב להיות מודול עצמאי עם אפשרויות שונות, כדי שיוכל להתאים לכל המפעלים של הלקוח.

ממשק המשימות

מאחר שסביר ביותר שמכונה תייצר מוצרים שונים במהלך מחזור החיים שלה, יש צורך שתתמוך במשימות שונות, המוגדרות על ידי המפעיל. בכל תעשייה ובכל מפעל יש כלים ותבניות ייחודיים להגדרת משימות אלו, והמשמעות היא שמפרש המשימות (task interpreter) חייב גם הוא להיות מודול עצמאי עם גרסאות מרובות על מנת שהמכונה תהיה שימושית בסביבות שונות.

המבנה של מכונה מודולרית

בחלק זה נתאר כמה מבין הדרכים הטובות ביותר במונחים של ארכיטקטורות של חומרה ותוכנה, אשר מאפשרות למכונות תעשייתיות להיות מודולריות באופן מלא.

בקר נשלט או תא חומרה

ההבדל בין בקר נשלט לתא חומרה (Hardware cell) הוא שבתוך הבקר הנשלט נכלל בקר עצמאי.

כפי שכבר צוין, על מנת לאפשר ללקוחות לבחור את התכונות הדרושות להם, יש לתכנן מכונות תעשייתיות עם תחנות נפרדות, אשר מבצעות תהליכים בדידים. כל אחת מתחנות אלו היא תא חומרה והיא נבנית כך שהיא מציגה ממשק פשוט בפני המכונה הראשית, בדרך כלל מחבר כניסה (IN) ויציאה (OUT) עבור fieldbus וחיבור אספקת הכוח. כך אפשר להרכיב את המכונה על ידי חיבור התאים הנפרדים יחד.

חלק מהתאים יכולים להיות בשימוש או לא, בתלות במשימה לביצוע, והמשמעות היא שאפשר לחזור ולהשתמש בהם במכונות שונות. כאמור, הן נקראות "יחידות נתקעות". אם אפשר להשתמש ביחידות אלו רק במכונות של יצרן מסוים, או אם יש בהן לוגיקה פשוטה מאוד, לא צריך לכלול בהן בקר, מאחר שהבקר של המכונה שאליו הן מחוברות ישמש לצורך הבקרה שלהן.

תאים מסוימים אחרים הם מורכבים יותר ואפשר להשתמש בהם במגוון מכונות, למשל במערכת טעינה רובוטית, אשר במקרה זה צריכה לכלול בקר נשלט. בקר נשלט כולל את הלוגיקה המיועדת לשליטה בתא שלו, אך אין היא מפעילה כל משימה או תוכנית של המשתמש, במקום זאת היא מקבלת את פקודות המשימה מהמחשב הראשי.

שכבת הפשטת חומרה

ללוגיקת הבקרה חייבת להיות דרך לפעול בגרסאות שונות של החומרה. כך אפשר להשתמש באותה הלוגיקה במחשבי המטען התכליתי (payload machine) וגם לשמור שהלוגיקה תהיה בלתי תלויה

בהתפתחויות חומרה. מאחר שהמכונה נבנית לשנים רבות, סביר להניח שיהיו שינויים מסוימים בחומרה. שמירה על לוגיקה בלתי תלויה תאפשר מימוש של תיקונים ועדכונים במכונות שנפרשו עם סוגי חומרה שונות.

כל זאת הופך כעת להיות לחשוב ביותר, מאחר שעדכונים אלו יכללו ככל הנראה עדכוני אבטחה במכונה ובממשק של המפעל. הכורח ליצור עדכון נפרד לכל גרסת חומרה היה מוביל למאמצים מיותרים וסביר להניח שמכונות ישנות לא היו יכולות לקבל עדכוני אבטחה ללא שכבת ההפשטה.

באופן כזה, מקבלים בוני המכונות יותר ביטחון ברכישה, מאחר שהם יכולים להחליף בין מותגי חומרה במקרה של בעיות אפשריות באספקה. מחסורים שהתרחשו לאחרונה בשוק המוליכים למחצה הפכו את הדרישה הזו לחשובה עבור בוני המכונות.

תוכנה מודולרית

בדיוק כמו תכונות החומרה, יש צורך לפתח את תכונות התוכנה בצורת מודולים, יש צורך שפלטפורמת התוכנה תוכל להתחבר אל אפיק fieldbus הפנימי ואל שכבות היישום השונות. בתוך הפלטפורמה, שכבת הפשטת החומרה מבצעת בקרה של אפיק fieldbus הפנימי, כדי לחשוף את תאי החומרה הזמינים אל היישומים.

לעתים נעשה פיתוח של יישומים שונים בשפות תכנות שונות וייתכן שתידרש להם פעולה בזמן אמת, ועל כן הפלטפורמה חייבת לפעול בזמן אמת. בה בעת, הממשק שלה חייב להיות חשוף לסביבות רבות ככל האפשר.

הפלטפורמה צריכה גם שיהיה לה קישור רופף ליישומים השונים כדי שתוכל להמשיך לפעול באופן תקין, גם אם חלק מהיישומים קורסים. פעולה זו קריטית מפני שחלק מהיישומים יכולים לכלול בטיחות. האפשרות שבה יישום אחד ישפיע על המערכת כולה יכולה לדרוש שכל יישום יהיה כפוף לכללי פיתוח בטיחותיים.

פלטפורמת התוכנה צריכה לאפשר תקשורת בין היישומים השונים, מפני שהמודולים צריכים לשתף זה את זה במידע. בדרך כלל, מצב עיבוד עכשווי של כל יחידה נשמר, כדי שכל מודול יתעדכן בו. כדי שכל המודולים יפעלו יחד באופן חלק, יישום ראשי יתחיל לפעול, יאתחל את הפלטפורמה ויפעיל את המודולים השונים, בהתבסס על קובצי קונפיגורציה ועל חומרה שמחוברים באותו זמן. הוא יבקר את כל התהליך הכרוך בהתחלה וכיבוי.

ממשק מכונה

התקשורת בין המכונה לבין שאר חלקי המפעל נחשבת כעת לתכונה חשובה ביותר של כל מכונה תעשייתית. עובדה זו נכונה לגבי כל המכונות, אך כאשר בונים מערכת מורכבת, היא יכולה להפוך להיות מבלבלת. ממשק המכונה חייב לאפשר גישה מרחוק אל החומרה לצורך תחזוקה ואבחונים, אל המידע הכולל של המכונה לצורך ניהול תהליכים וניתוחים, ואל התכונות השונות כדי לאפשר שיתוף פעולה עם מכונות שונות.

כדי להקטין את ההתקפות האפשריות נגד המכונה ולהימנע מהפרעות אקראיות לייצור, אסור שהגישה אל החומרה תהיה ישירה, הגישה לתחזוקה ולאבחונים חייבת לעבור דרך בקר במכונה. באופן כזה אפשר להבטיח שהמכונה נמצאת במצב תחזוקה ושהקשר בין הבקר לחומרה לא ינותק. ממשק זה משתמש במודול פרוטוקול ותוכנה שונה מזה שבו משתמשים ממשקי המכונה האחרים, מאחר שהוא חייב להיות מאובטח ברמה עליונה ולאפשר בקרה מרחוק במקום גישה דרך משתנה וגישה דרך ממשק API.

הממשק לניהול התהליך ולניתוחים בדרך כלל פשוט ביותר, מפני שהוא אוסף נתונים באופן מחזורי. ואולם, כדי שהניתוחים והניהול יהיו שימושיים, חייבת להיות אפשרות לפעול על פי התוצאות, ולכן גם אפשרות לשנות את סדרי העדיפות, את לוחות הזמנים ואת הכיול של המכונה. בנוסף לגישה תקופתית אל נתונים שנקבעו לפי הקשר, ממשק זה צריך שיהיו לו ממשקי API שיעדכנו את המכונה.

המורכבות של הממשק המשמש לשיתוף פעולה של המכונה תלויה באופן נרחב במידה שבה המכונות תלויות אלו באלו, שוב, תלות שתלויה בתרחיש הפרישה. במקום הצורך לבנות ממשק שונה לכל פרויקט, תקנים שנקבעו לאחרונה מציעים למזג את הממשק הזה עם ממשק הניהול והניתוחים ולספק תיאור מלא של תכונות המכונה לרבות המצב, שבו אפשר לשנות פרמטרים, ושל הפונקציות שלהן אפשר לקרוא. כדי שכל אלו יתקיימו באופן בטוח, ייתכן שיחולו מגבלות גישה שונות על תכונות ופרמטרים שונים. כל אחת מהמערכות במפעל, כגון מערכת SCADA, מערכת MES, ניתוחים ומכונות אחרות, ישתמשו בהרשאות אחרות ותהיה להן גישה לתכונות שונות שנדרשות להן. לצורך השילוב, כעת כבר אין צורך לערוך שינוי בממשק המכונה, רק בהגדרה של ההרשאות השונות.

קובץ המשימות

ככל שמכונות משנות כיום את המשימות, הן פועלות לעתים קרובות מאוד, ויכולות אפילו לבצע משימות שונות על פריטים שונים, באותו זמן. אם משימות נמצאות בתחנות שונות, הן לא יכולות לדרוש הגדרות ידניות על ידי מפעיל.

באופן דומה למכונות CNC עם קובצי קוד G/M, יש לאחסן את הגדרת המשימה בקובץ שניתן בקלות להחליפו, לשנות אותו או להוריד אותו משרתים בעקבות הפקודות של מערכת התזמון של המפעל. במקרים מסוימים, במקום שתהיה מערכת תזמון שמקצה משימות לכל מכונה, היא מקצה משימות לפריטים באמצעות תג זיהוי, והמכונות יורידו באופן אוטומטי את קובץ המשימה שלהם לאחר קריאת תג הזיהוי של הפריט הבא.

דרישות טכניות

על מנת לקבל את הארכיטקטורה הנדרשת על ידי המכונה המודולרית, לחומרה, לתוכנה ולפרוטוקולים המשמשים את המכונה יש צורך בתכונות מסוימות.

fieldbus עם חיבור חם, סריקת אפיק וביצוע קונפיגורציה אוטומטי

הגורם בעל החשיבות העליונה הראשון שיש לבחור בו הוא אפיק fieldbus המשמש לחיבור הבקר עם חלקי החומרה השונים. מאחר שלמכונות מודולריות יכולות להיות

אפשרויות רבות, אפיק fieldbus צריך לאפשר חיבור של יותר ממאה התקנים עם מחזורים של פחות מ-1 מילי שנייה, מאחר שלחלק מהמרכיבים יש צורך בקצבי דגימה גבוהים.

האפיק צריך גם לאפשר חיבור בשרשרת ותיוג של החומרה, כדי שאפשר יהיה לחבר תאי חומרה בגב כל אחד מהם ולזהות אותם גם כאשר חומרה זהה משמשת פעמים רבות. מאחר שיש אפשרות לשילובים רבים, יש צורך שאפיק fieldbus יאפשר לסרוק את פריטי החומרה המחוברים עם האתחול, כדי שהבקר יותאם באופן אוטומטי למה שמחובר אליו.

כפי שכבר צוין, יש צורך שהתחזוקה ואבחונים יהיו אפשריים לביצוע מרחוק דרך הבקר, כך שאפיק fieldbus צריך לאפשר הגדרת קונפיגורציה, אבחונים ועדכון של ההתקן. אם יש צורך לבצע תחזוקה או שינוי במכונה תוך שהיא פועלת, למשל אם חלקים מהתהליך נמשכים זמן רב, צריך שאפיק fieldbus יתמוך בחיבור חם והמשמעות היא הסרה או הוספה של חומרה תוך כדי פעולה.

מאחר שבמכונות אלו נעשה שימוש בסוגי חומרה שונים מאוד, יש צורך שהפרוטוקול יאומץ על ידי ספקי חומרה רבים. כך גם יעמדו לרשות בוני מכונות אפשרויות חומרה מרובות במקרים של בעיות באספקה.

מחשב PC סטנדרטי

היחידה החשובה הבאה היא חומרת הבקר. בקרים מודרניים כוללים הן רכיבי זמן אמת וגם רכיבים שאינם רכיבי זמן אמת, והמשמעות היא שהחומרה חייבת להיות יע״מ (CPU), עם ריבוי ליבות ועם חיבורי port לרשת. במקרים של נפחים גדולים, מומלץ בדרך כלל להשתמש במערכת SoC שמתוכננת במיוחד עבור הבקר. שימוש כזה עלול לעתים להיות לא יעיל, למשל כאשר הבקר משמש במכונות רבות ושונות. במקרה כזה מערכת SoC צריכה לתמוך בגרסה המורכבת ביותר של המכונה, או שיהיה צורך במערכות SoC רבות ושונות, כך שהנפח לכל מערכת SoC יקטן מאוד וזמן הפיתוח של כל מכונה חדשה יגדל.

עבור מכונות מודולריות, מחשב PC סטנדרטי הוא המתאים ביותר מאחר שיחידות יע״מ שונות וזיכרונות RAM שונים יכולים לשמש בכל גרסה של המכונה, מבלי שיהיה צורך בעבודה נוספת או בעלויות תכנון נוספות. מחשבים סטנדרטיים מאפשרים שימוש במעגלי PCI סטנדרטיים, עובדה שהיא חשובה לתקשורת של המפעל.

מערכות משנה עם ממשקים מרובים

על מנת לבנות תוכנה מודולרית, חשוב ביותר שתהיה תת מערכת מרכזית שתבצע חיבורים פנימיים של כל המרכיבים. אפשרות נוספת היא ליצור פירמידה של המודולים השונים, אשר יכולים לקרוא אחד לשני, אבל ארכיטקטורה מסוג זה אינה גמישה מפני שלא ניתן להוסיף בקלות מודולים באמצע המערום הקיים.

המלצות רגילות של תכנון תוכנה יכולות להיות למשל שהממשק יטפל רק בחיבורים הפנימיים שבין המודולים, אבל מצב זה עלול להוסיף מורכבות לפיתוח, מאחר שחלק מהפונקציות (fieldbus ותנועה) משמשות כמעט את כל מודולי היישומים. לכן מומלץ לפתח תת מערכת כפלטפורמת יישומים אשר תכלול את אפיק fieldbus, גישה לכניסות וליציאות ותנועה סטנדרטית, נוסף לחיבורים הפנימיים של היישום. מאחר שחלק ממודולי היישומים יהיו לזמן אמת, תת מערכת זו צריכה לפעול גם בצד של זמן אמת של הבקר, והמשמעות היא שלא תוכל לכלול את היישום הראשי, אשר פועל בצד הבקר שאינו לזמן אמת. לארכיטקטורת התוכנה יהיה יישום ראשי אשר יפעיל את תת המערכת של הבקרה שכוללת את אפיק fieldbus, לאחר מכן אפיק fieldbus יתחיל לפעול ויסרוק את האפיק והיישום הראשי יפעיל את מודולי התוכנה הנוספים בהתבסס על קובצי החומרה והקונפיגורציה שנסרקו.

מאחר שמודולי תוכנה אחרים יכולים להיות שונים מאוד, החל בתהליך זמן אמת ועד כלי החוזי (visualization) או ניתוחים של נתונים, אפשר לפתח אותם בשפות ובסביבות שונות ורבות, לכן תת המערכת צריכה שיהיו לה ממשקים עבור סביבות זמן אמת תעשייתיות כגון C, C++ ו- PLC וסביבות כגון .NET, Java או Python.

תקשורת ובקרה סטנדרטיות

כאשר מתבצעת פרישה של מכונה, לעתים קרובות מאוד היא נוספת לקו ייצור קיים במפעל, כך שצריך לחבר אותה בממשק עם המכונות הקיימות ומערכת האוטומציה של המפעל. המשמעות היא שיש צורך בתמיכה בטווח רחב של פרוטוקולים סטנדרטיים וקנייניים. לכן, חייבת להיות אפשרות להוסיף כרטיס תקשורת של פרוטוקול לפי בחירה, בדרך כלל כרטיס PCI, על מנת להתממשק עם המפעל ועם מכונות אחרות. כדי לבצע זאת מבלי לשנות את תוכנת הבקר, הממשק של המכונה צריך להיות מבוסס על תקן פתוח שנמצא בשימוש נרחב, כך שיהיה פשוט לאתר רכיבים שמבצעים המרה מהפרוטוקול הסטנדרטי לפרוטוקול של המפעל.

בנוסף, סביר ביותר להניח שכל לקוח יבקש ביצוע של שינויים קלים בלוגיקה או שיתאפשר לו להוסיף את לוגיקת התוכנה שלו. ברוב המקרים לוגיקה זו תשמש לבקרה ולסנכרון של הממשק עם מכונות אחרות. כדי להימנע מהצורך לשלוח מהנדסים לביצוע שינויים בבקר בכל פרויקט, הבקר עצמו צריך לכלול סביבת בקרה סטנדרטית שתאפשר ללקוח להוסיף בעצמו את הלוגיקה, וסביבה זו היא בדרך כלל בקר PLC בתוכנה.

משימה לפי תסריט

מאחר שמכונות מבצעות כיום משימות רבות ושונות באותה העת, יש להגדיר את המשימות האלה בקבצים שקל לשנות ולעדכן אותם. במקרים מסוימים, צריך לערוך את המשימות תוך כד ביצוע. במקום להשתמש ביישום שמצריך איסוף (compiled), מקבלים גמישות רבה יותר בשימוש בשפה ליצירת תסריטים (scripting) ולאספקת יישום עריכה (editor) ישירות בבקר.

בקרה ותחזוקה מרחוק

לבסוף, חשוב מאוד לספק גישה מרחוק אל המכונה כדי לאפשר ביצוע של בקרה ואבחון, להיות מוכנים לתחזוקה ולהגדיר במדויק את המשימות שיבוצעו על ידי המכונה. כתוצאה, המהנדסים המקומיים

יכולים לבצע את התחזוקה במכונה ללא הדרכה מיוחדת.

KINGSTAR

לאחר שסקרנו את הסוגים השונים של מודולריות ודנו בפעולות הטובות ביותר ובדרישות הטכניות לבניית מכונה מודולרית, נבדוק את הפתרון של KINGSTAR.

כפי שצוין קודם, KINGSTAR היא פלטפורמת תוכנה לאוטומציה של מכונות. היא מיועדת ליצרנים של בקרים ולבוני מכונות. KINGSTAR היא חטיבה של IntervalZero, חברה שבסיסה בארה"ב ויש לה משרדים ברחבי העולם, אשר מתמחה מזה שנים רבות במערכות זמן אמת ומערכות משובצות.

בעבר IntervalZero הייתה חברה שהתמקדה בבקרי מכונות מבוססי Windows.

חברת IntervalZero פיתחה ותחזקה את קו מוצרי RTX במשך יותר מעשרים שנה, כשהיא מספקת הרחבה של מערכת Windows לזמן אמת. RTX משמשת עם לוחות מעגלי בקרה ותקשורת ליצירת בקרי מכונות. עם העלייה בכוח המעבדים וההופעה של פרוטוקולי fieldbus מבוססי תקשורת Ethernet, הלקוחות ממשיכים לדרוש פרוטוקולי תוכנה ולוגיקת בקרה מבוססת תוכנה. לכן חברת IntervalZero פיתחה את הפלטפורמה של תוכנת האוטומציה KINGSTAR המיועדת לבניית בקרי מכונות חכמים.

KINGSTAR מורכבת מחמישה רכיבים:

- **KINGSTAR Fieldbus** (Master EtherCAT® לזמן אמת)
- **KINGSTAR Motion** (בקרת תנועה)
- **KINGSTAR PLC** (בקר תוכנה לוגי מתוכנת)
- **KINGSTAR Vision** (פתרון GigE Vision לזמן אמת)
- **KINGSTAR IoT** (פלטפורמה מאופשרת האינטרנט של הדברים – IoT)

בקרת זמן אמת מבוססת מחשב PC

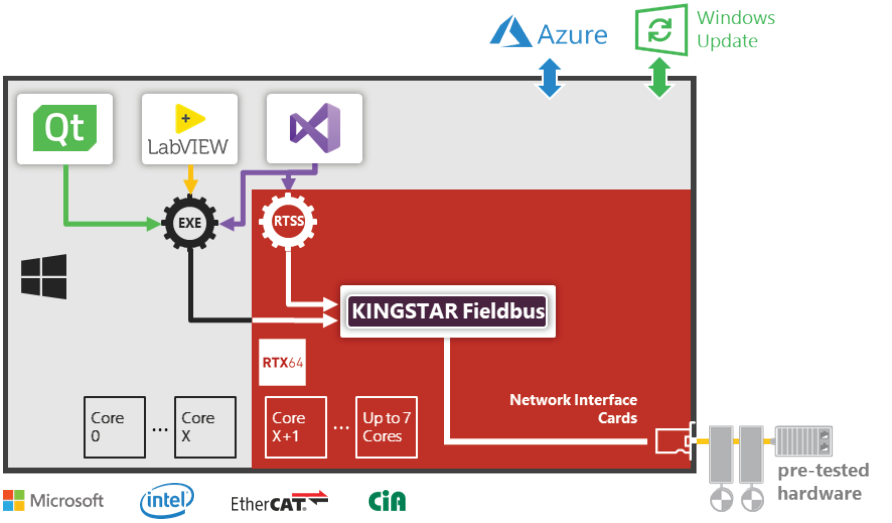
ההיבט החשוב הראשון הוא פלטפורמת התוכנה וכפי שהוסבר קודם, מחשב PC סטנדרטי עם מערכת הפעלה ברמה עולמית מספק יתרונות רבים.

KINGSTAR פועלת בגרסת 64 הסיביות של ההרחבה לזמן אמת, RTX64, שהיא רכיב הליבה של פלטפורמת KINGSTAR והיא הופכת את Windows למערכת הפעלה הפועלת בזמן אמת.

הפעלה ב- RTX64 ל- 64 סיביות במערכת ההפעלה Windows 10, מאפשרת פיתוח של יישומים לזמן אמת עם C/C++ ב- Visual Studio. אפשר להשתמש בה בטווח רחב של מחשבים לשימוש כללי והיא פרושה בתעשיות שונות ורבות, כגון במערכות אוטומציה ורובוטיקה, אבל גם במערכות רפואיות, צבאיות ובמערכות מדמה (simulator).

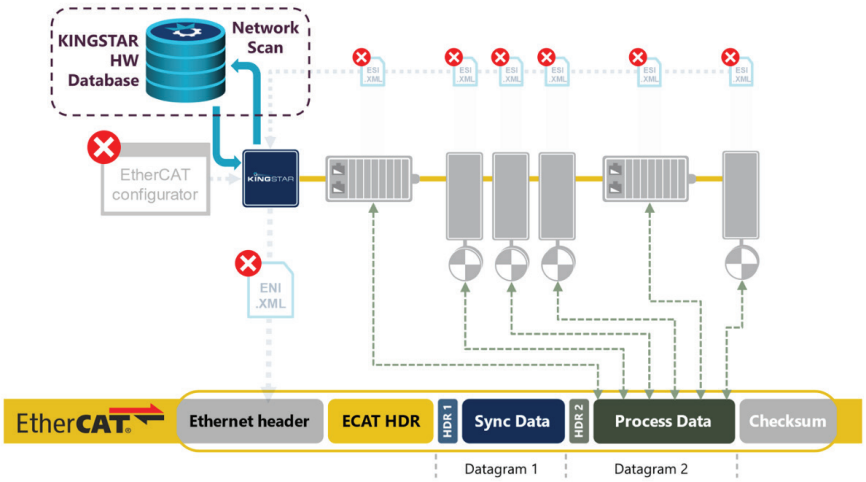
EtherCAT בהגדרה עצמית

אפיק fieldbus שמשמש ב- KINGSTAR מממש מערום 'הכנס והפעל' EtherCAT® ב- RTX64. כפי שצוין בעלון המידע אשר משווה את חמשת אפיקי fieldbus החשובים ביותר שבשוק, IntervalZero מאמינה שמערכת EtherCAT® היא הפרוטוקול הטוב ביותר לאוטומציה של מכונות, ועליה מבוסס המוצר KINGSTAR. על מנת לספק גמישות רבה יותר ליישומים,



איור 1: ארכיטקטורת Fieldbus של KINGSTAR

IntervalZero ניצלה את יכולות סריקת האפיק של EtherCAT® כדי לבנות תכונת ביצוע אוטומטי של קונפיגורציה, וכך אותו יישום יכול לפעול בקונפיגורציות שונות של חומרה. היתרון העיקרי של ביצוע קונפיגורציה אוטומטי זה הוא התמיכה בכל הינעי הסרוו העיקריים וכל מותגי חומרת קלט/ פלט, והמשתמשים יכולים להוסיף תמיכה בחומרה חדשה מבוססת EtherCAT מבלי שיצטרכו לעדכן את KINGSTAR. בנוסף, שכבת fieldbus מספקת גישה ישירה למשתנים כאילו היו מקומיים, כשהיא מסתירה את אפיק fieldbus מהיישומים.



איור 2: תכונת ביצוע הקונפיגורציה האוטומטי של KINGSTAR

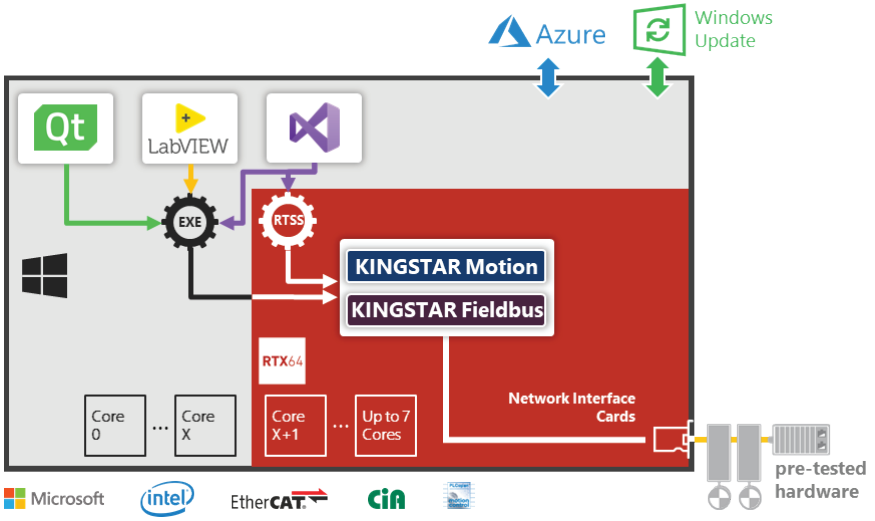
מבוססת על תקנים: EtherCAT, CAN, DS402, PLCopen, OPC UA

על מנת להשלים את הפלטפורמה לבקרת מכונות חכמה, מספקת KINGSTAR גם רכיב תנועה בתוכנה. KINGSTAR Motion תואמת למפרט תקן PLCopen Motion Control עבור בקרת תנועה שמשמשת בתנועה מסונכרנת של קבוצה (group motion) מנקודה לנקודה, מיזוגוקינמטיקה. עם מעבדים מודרניים ומשוואות התנועה המותאמות באופטימיזציה בפלטפורמת KINGSTAR, אפשר לבצע בקרה של מספר צירים גדול בזמני מחזור מהירים. לדוגמה, אפשר להשתמש ביישומים עם 20 צירים ועם זמני מחזור של 125 מיקרו שנייה או עם 60 צירים עם זמני מחזור של 500 מיקרו שנייה. בכל ציר אפשר להשתמש במותג חומרה אחר והוא יכול שיהיה לו מצב בקרה משלו. התקשורת עם ההינעים מבוססת על מצבים מחזוריים סינכרוניים, האינטרפולציה מתבצעת בבקר, אך ביצוע נגזרת PID יכול להיות בבקר או בהינע. אלגוריתמי התנועה מאפשרים עריכת שינויים בפרופיל התנועה תוך כדי תנועת הציר. הסנכרון תומך בהינע זיזים אלקטרוני (electronic camming), בהינע תמסורת (gearing) ובתנועה קבוצתית עם תנועות ליניאריות, מעגליות ובתנועות סליל (helical). תכונות התנועה האלו של KINGSTAR Motion גמישות ביותר מאחר שלמנגנון CAM או לציר שולט (master) בתמסורת יכולים להיות נשלטים רבים, והוא עצמו יכול להיות ציר וירטואלי או אף ציר נשלט על יד ציר אחר. תכונות תנועה אלו קיימות ביישומי זמן אמת וגם ביישומי Windows.

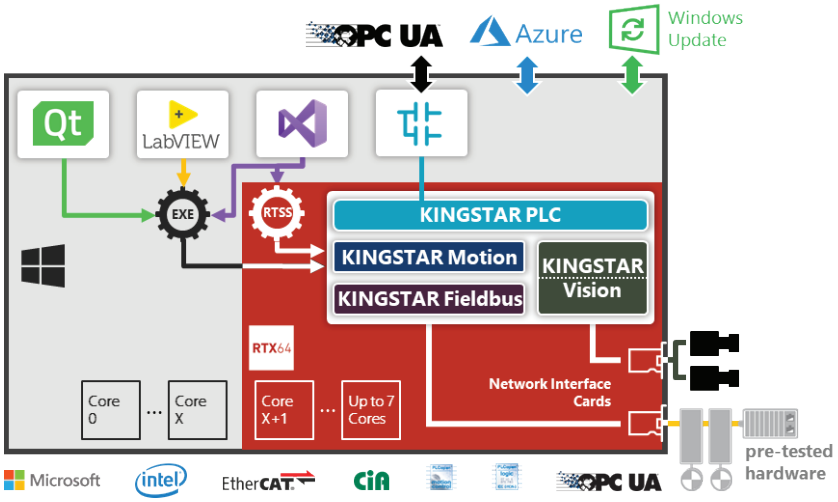
הרכיב השלישי הוא KINGSTAR PLC אשר מספק תוכנה מבוססת בקר PLC עם תכונות מלאות ומשולבת על גבי מערכת הפעלה לזמן אמת (RTOS) מבית RTX64 – IntervalZero. הבקר KINGSTAR PLC כולל גם רכיבי תוספת או רכיבים של צד שלישי לבקרת התנועה ולראייה מכונה אשר מנוהלים באמצעות ממשק משתמש עשיר עבור מתכנתים בשפת C++ ובה במידה עבור מי שאינם עוסקים בפיתוח.

KINGSTAR Vision היא מחסנית של ממשק GigE Vision® בזמן אמת, אשר מאפשרת ללקוחות לפתח בקרת תנועה בהכוונת ראייה באמצעות OpenCV (ספריות מקור פתוח) על מחשב PC עם Windows. המחסנית KINGSTAR Vision היא אוסף מקיף של כלי תוכנה לפיתוח יישומי ראיית מכונה, ניתוח תמונה ודימות רפואי על ממשק GigE Vision® וממשקי מצלמות אחרים. היא כוללת כלים עבור כל צעד בתהליך, מבדיקת היתכנות יישומים, ליצירת אבי טיפוס, עד פיתוח ופרישה סופית.

לבסוף אבל לא פחות חשוב, KINGSTAR



איור 3: הארכיטקטורה של KINGSTAR Motion



איור 4: KINGSTAR PLC עם ארכיטקטורת KINGSTAR Vision

IoT עבור מחשבי PC עם Windows מוסיפה לבקרת המכונה שלך פונקציונליות לאינטרנט של הדברים (IoT) על ידי מימוש פלטפורמת התוכנה הפתוחה לאוטומציה של מכונות. לקבלת מידע נוסף בנושא זה, קיימים כמה עלוני מידע באתר האינטרנט שלנו בכתובת www.kingstar.com. במיוחד עלון המידע שלנו Achieving Industry 4.0: Foue Critical Features for Smart Machine Automation (שימוש ב-Industry 4/0: ארבע תכונות חיוניות לאוטומציית מכונות חכמה) מציג ניתוח מעמיק בנושא זה.

פלטפורמה שניתנת להתאמה אישית

בנוסף למודולריות, דרישת השוק העליונה היא היכולת לבצע התאמה אישית בבקרים או במכונות. כבר חקרנו את הנושא הזה בעלון מידע אחר ובסמינר ברשת. כמה מבין הדרישות הטכניות דומות ביותר לאלו שנדרשות עבור מודולריות. פלטפורמת תוכנה לאוטומציה של מכונות חייבת להיות פתוחה ולתמוך בתקנים. מכונות ובקרים הופכים מורכבים יותר ויותר, ולפיתוח התכונות שלהם נדרשים צוותים מרובים עם כישורים שונים. לפיתוח של אלגוריתם תנועה ייעודי בליבת הבקר יש צורך בכישורים של זמן אמת ותכנות בשפת C++ כאשר לממשק משתמש גרפי בהתאמה אישית עבור המפעיל נדרשים כישורי ממשק HMI בתכנות של סביבת בקר PLC. בינתיים, שפות וסביבות אחרות יכולות להיות אפשרות טובה יותר, כמו למשל .NET. לממשקי משתמש עבור אינטגרטורים של מערכות.

מסקנות

KINGSTAR תוכננה מתוך התחשבות בתקנים פתוחים ודרישות השוק. מודולריות היא כנראה אחת הדרישות הגדולות ביותר מצד יצרני בקרים וגם מצד בוני מכונות. דרישות נוספות, כגון התאמה אישית, תחזוקה ועזרה מרחוק או אבטחה, גם הן דרישות חזקות, ואלו כבר נדונו בעלוני מידע אחרים או סמינרי רשת, כפי שכבר צוין קודם, או שהם יהיו הנושאים של עלוני מידע אחרים בעתיד. נקודה נוספת שיש לציין אותה היא היכולת של KINGSTAR להתחבר לעידן של Industry 4.0. רוב בוני המכונות ומכאן גם יצרני הבקרים חייבים לקחת בחשבון את Industry 4.0 בתכנון שלהם. שוב, תמיכה בתקנים כגון בקרים מבוססי מחשב PC, מערכת ההפעלה Windows 10, OPC, UA, AZURE וכיו"ב היא השיטה הטובה ביותר להכין את הדור הבא של המערכות שלך, בין אם בקר או מכונה שכוללים שירותי האינטרנט של הדברים.