From Benchmarking to Prediction: Energy Profiling of Industrial Automation Systems using Machine Learning

Dimitris Kallis **Moysis Symeonides** **Marios D. Dikaiakos**

Laboratory for Internet Computing

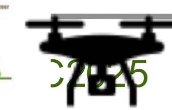
Dept. of Computer Science

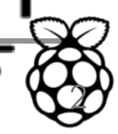
University of Cyprus

IEEE ISCC2025, Bologna, Italy

Cyber-physical Systems (CPS)

- Integrate computation with physical processes to enable intelligent, real-time control
- Key features:
  - Embedded devices monitor and control physical processes
  - Feedback loops ensure physical processes and computations dynamically influence each other
  - Multi-scale and heterogeneous components blend diverse technologies and levels of abstraction





CPS in Manufacturing

- In industrial settings, CPS deployments feature integrated **sensing**, **actuation**, and **computation** to enable complex, autonomous task execution (**IIOT**).
- **Trends** Driving Adoption
  - Acceleration of automation across manufacturing sectors
  - Robotics price reduction accelerates mass deployment: robotics arms dropped 46% (2017–2021)
  - Projected boost in global productivity and economic growth
- Some **Challenges**
  - Power requirements are a key operational cost of IIoT systems
  - Essential to evaluate energy, power, and performance, considering both computational and physical components used under different workloads
- **Gap**
  - Existing works typically rely on simplified models or narrow case studies, lacking system-wide evaluations of energy use in realistic IIoT deployments.

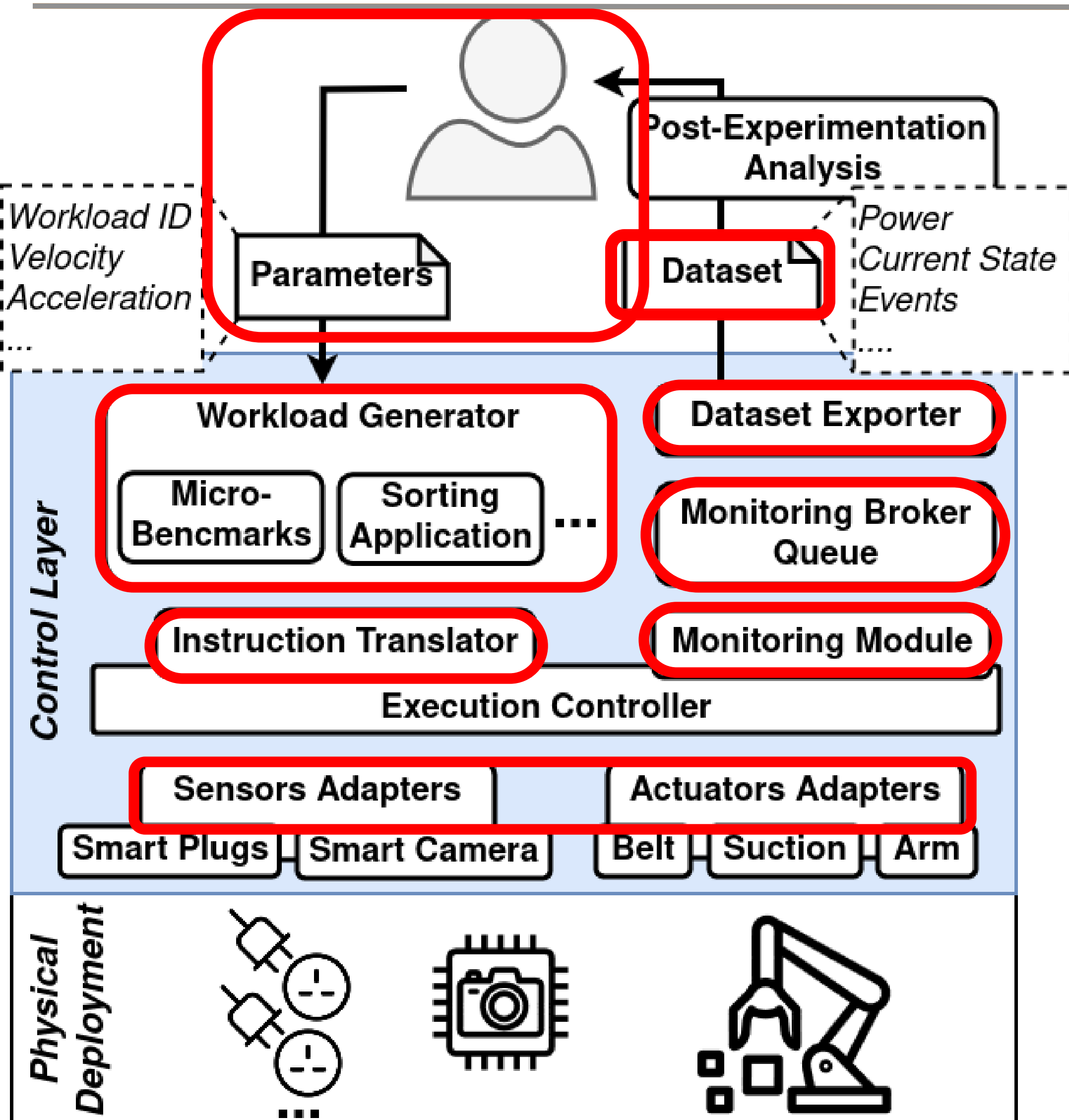
Contribution Overview

- A methodology and a parameterized benchmarking framework to
  - profile the energy consumption of industrial CPS configurations
  - develop predictive models of their power and energy requirements on different workloads
- Framework validation on a realistic automated object-sorting system, comprising robotic arm, conveyor belt, smart camera and application software
- Monitoring system to profile energy usage at both the physical (actuators) and digital (sensing/computation) layers
- Mapping energy consumption to physical operational parameters (e.g., speed, weight, acceleration)
- Training predictive ML models to estimate end-to-end application energy profiles based on task features
- Evaluating energy prediction models and features

Architecture Overview

- Modular Design:
  - **Control Layer** abstracts physical APIs, enabling repeatable experiments, workload extensibility, and dataset extraction
  - **Workload Generator** loads and configures application scenarios parameterized with user-defined parameters (e.g., arm speed, belt acceleration)
- Execution Pipeline:
  - **Instruction Translator** converts high-level workloads into actuator-readable commands
  - **Execution Controller** orchestrates task execution and coordinates sensor/actuator adapters
  - **Actuator Adapters** for robotic arm, belt, and suction end-effector (via Dobot API)
  - **Sensor Adapters** for smart camera (object detection) and smart plug (power monitoring)
- Data Collection & Monitoring:
  - **Monitoring Module** logs commands, statuses, and runtime metrics with timestamps
  - **Monitoring Broker Queue** (RabbitMQ) disseminates metrics asynchronously
  - **Data Exporter** listens to queues and compiles trial-level datasets in CSV format, capturing second-by-second system states and energy data

Framework Operation



Step 1: **User** submits to the control layer a set of experiment parameters, which define the selected workload and configuration preferences.

Step 2: **Workload Generator** loads the code of the selected application scenario and configures the system.

Step 3: Execution Controller invokes **Instruction Translator** to translate the workload specification into instructions readable from physical components of the infrastructure.

Step 4: **Execution Controller** executes the workload on the physical infrastructure, by invoking calls to the **Sensor** and **Actuator Adapters**, which trigger operations of the sensing and actuator devices.

Step 5: Execution Controller uses the **Monitoring Module** to log and timestamp every command dispatched to the physical setup and the status of the physical infrastructure and disseminate traces to the **Monitoring Broker Queue**.

Step 6: **Data Exporter** listens to Monitoring Broker Queue and appends new data points received to the monitoring dataset.

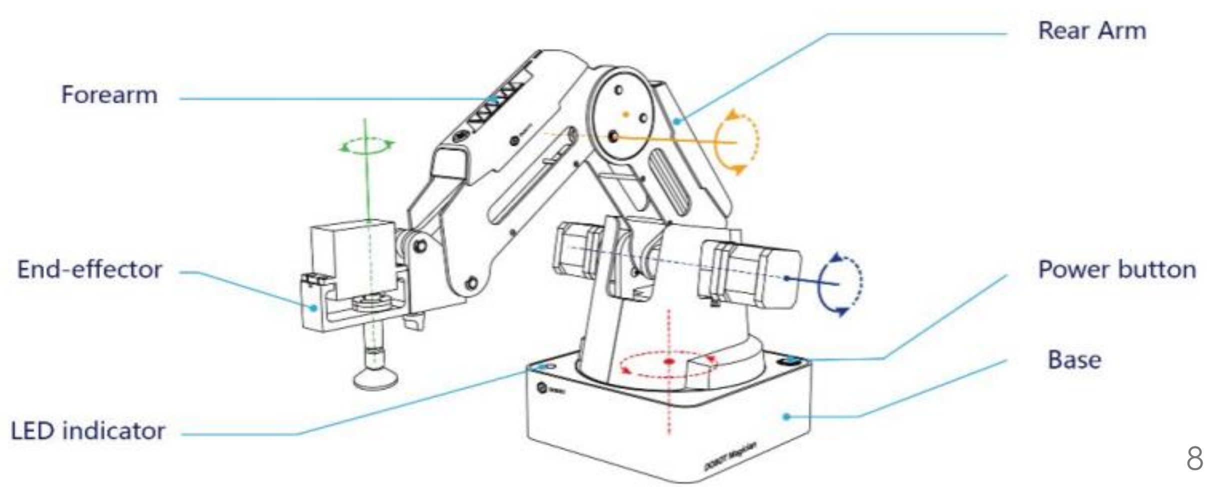
Equipment: Actuators

Robotic Arm: Dobot Magician

- World's first desktop grade 4-axis robotic arm: interchangeable end-effectors
- Motion control includes joint-based and Cartesian movements with programmable velocity and acceleration
- Graphical & script-based programming, full-featured API
- Performance: up to 320°/s rotation for arm components and 480°/s for the end-effector servo with a 250g load

Conveyor belt kit:

- Supports small-scale c



Equipment: Sensors

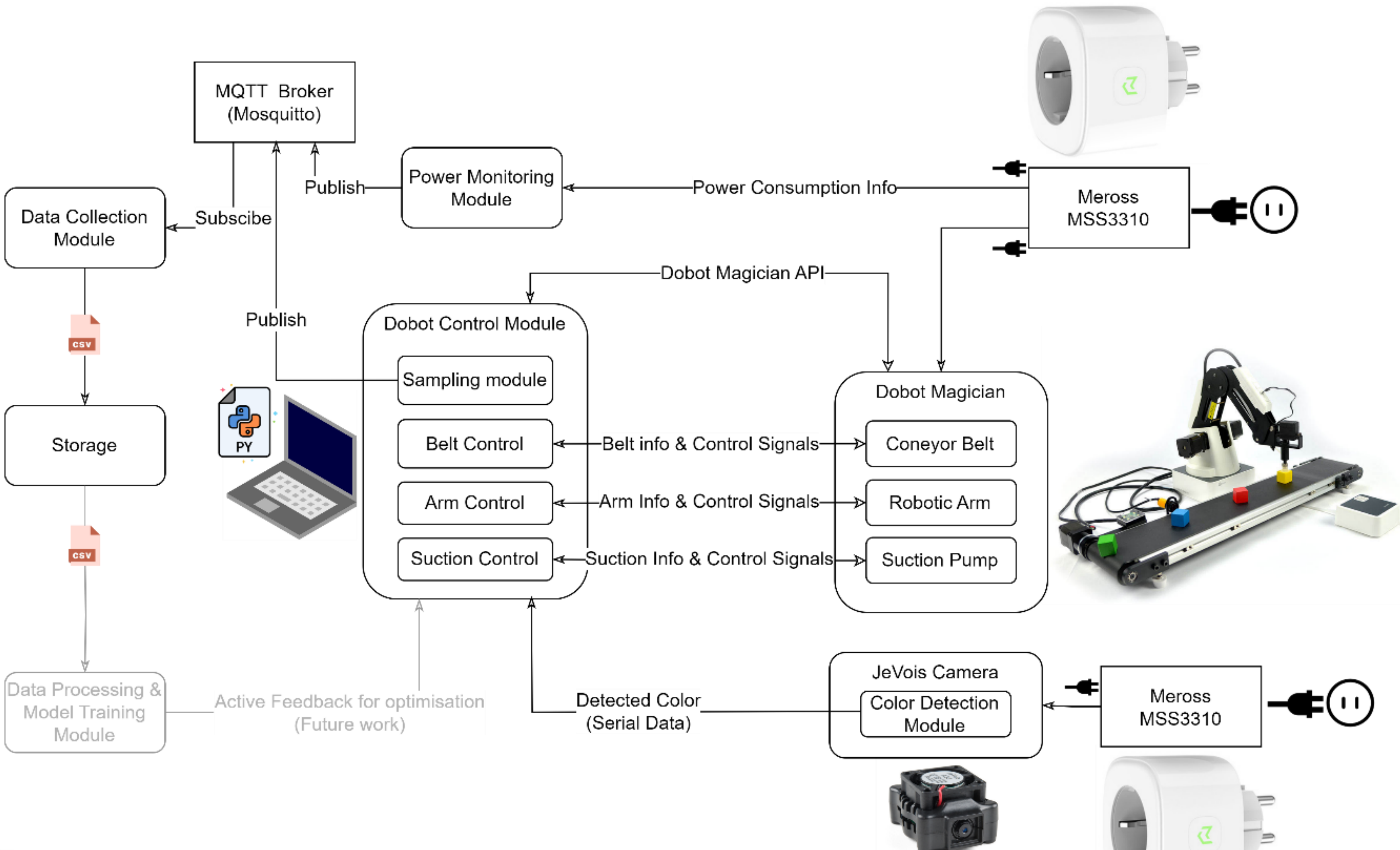
Embedded camera JeVois-A33

- Camera, embedded quad-core processor, and USB video interface
- ML-based vision tasks, supports OpenCV, TensorFlow, and Caffe
- Functions as a plug-and-play USB webcam w/ serial msg com.
- Outputs include object identity, 3D location, color detection, and counting

Meross smart plug: monitors energy consumption of individual subcomponents.

- Each actuator is connected to a separate smart plug.
- Provide network-accessible API for data collection and of power consumption (e.g., watts, voltage, amperage), supporting automated, programmatic real-time retrieval over Wi-Fi

Experimental Setup



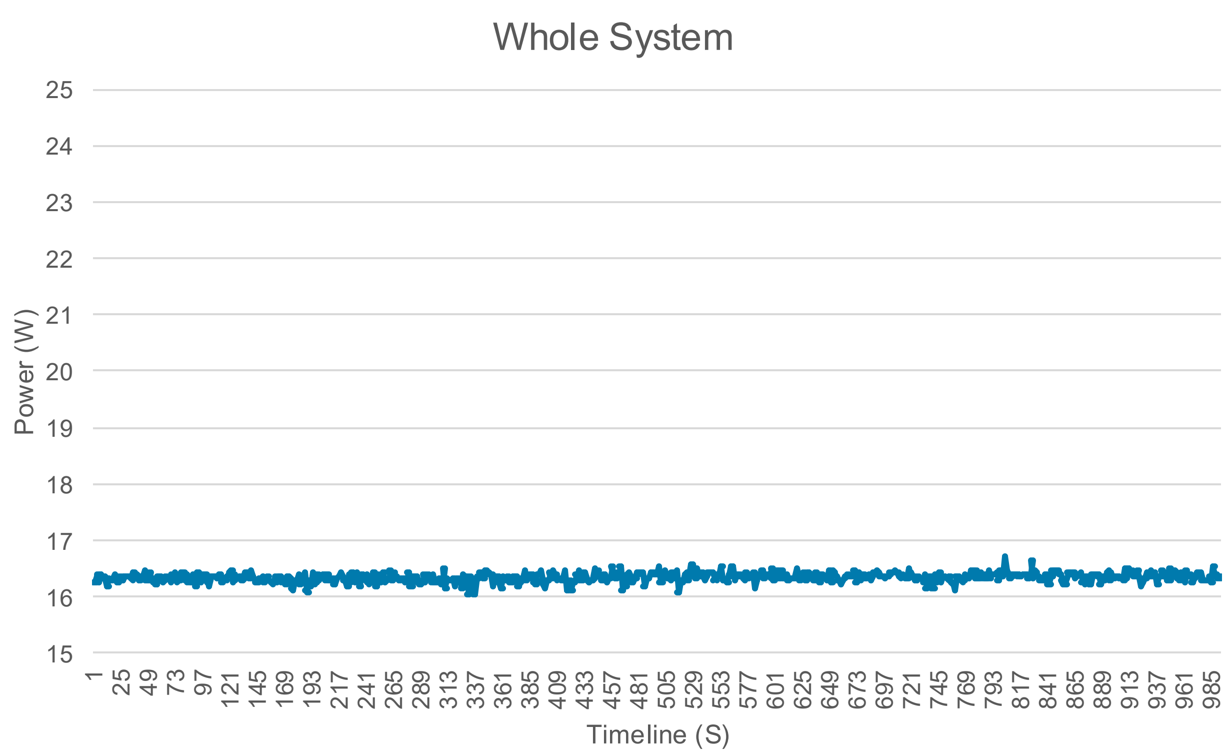
Experimental Setup

Workloads: Microbenchmarks

- Aim: Component-Level Energy Evaluation
- Simple tasks dedicated to each physical component:
- **Robotic Arm**: Move between position A and B with configurable speed and acceleration.
- **Camera**: Run color detection with two states:
  - Object detected → color identified
  - No object → no color identified
- **Conveyor Belt**: Start/stop movement w/ adjustable speed.
- **Suction End-Effector**: Enable or disable suction.
- **Payload Testing**: Manually vary object weight (up to 730g).

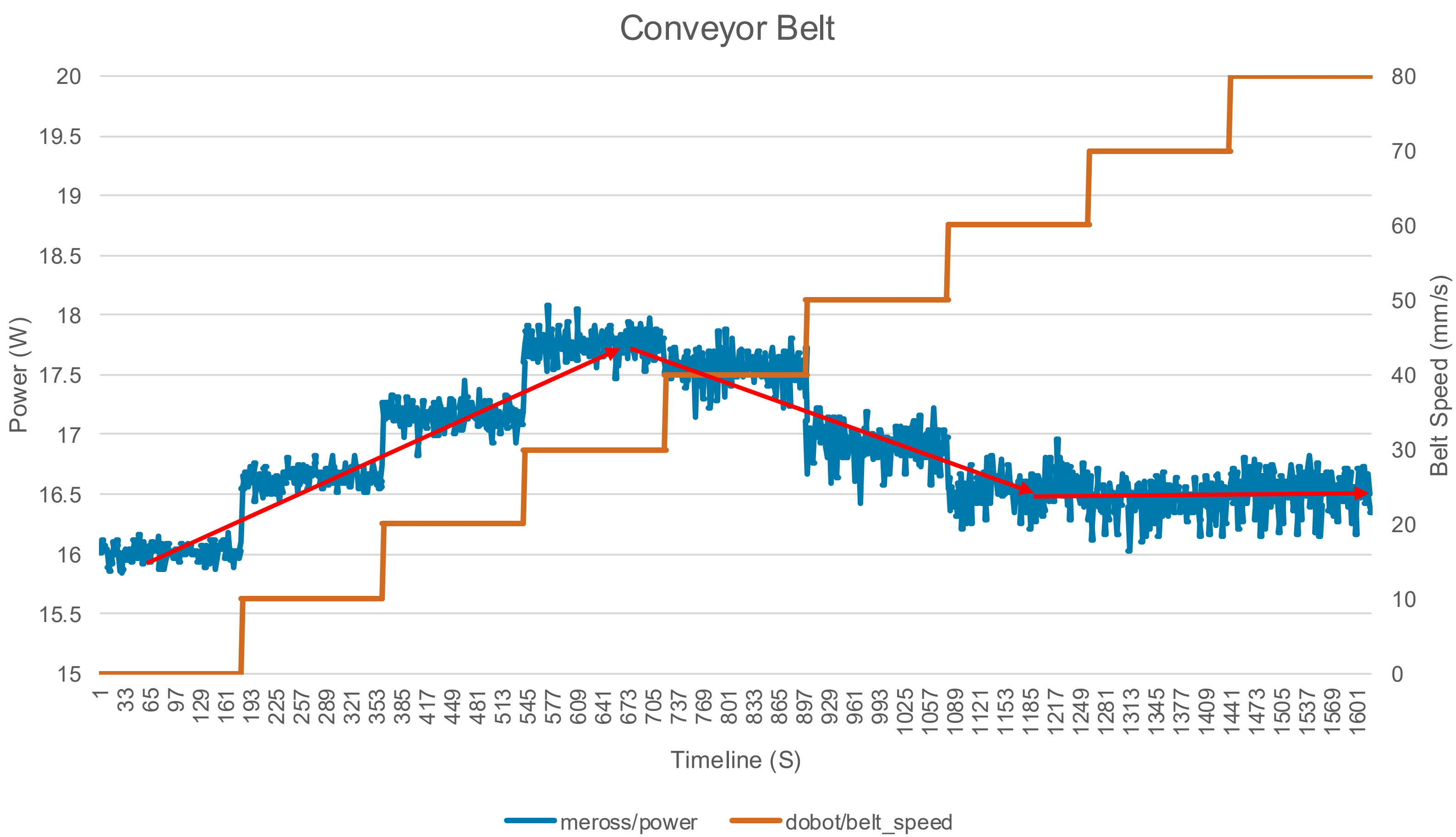
Exploratory Study

- System in idle state
- Range: 16.044W - 16.709W
- Average: 16.339W; Median: 16.348W
- After a long time of idle state power consumption remains stable

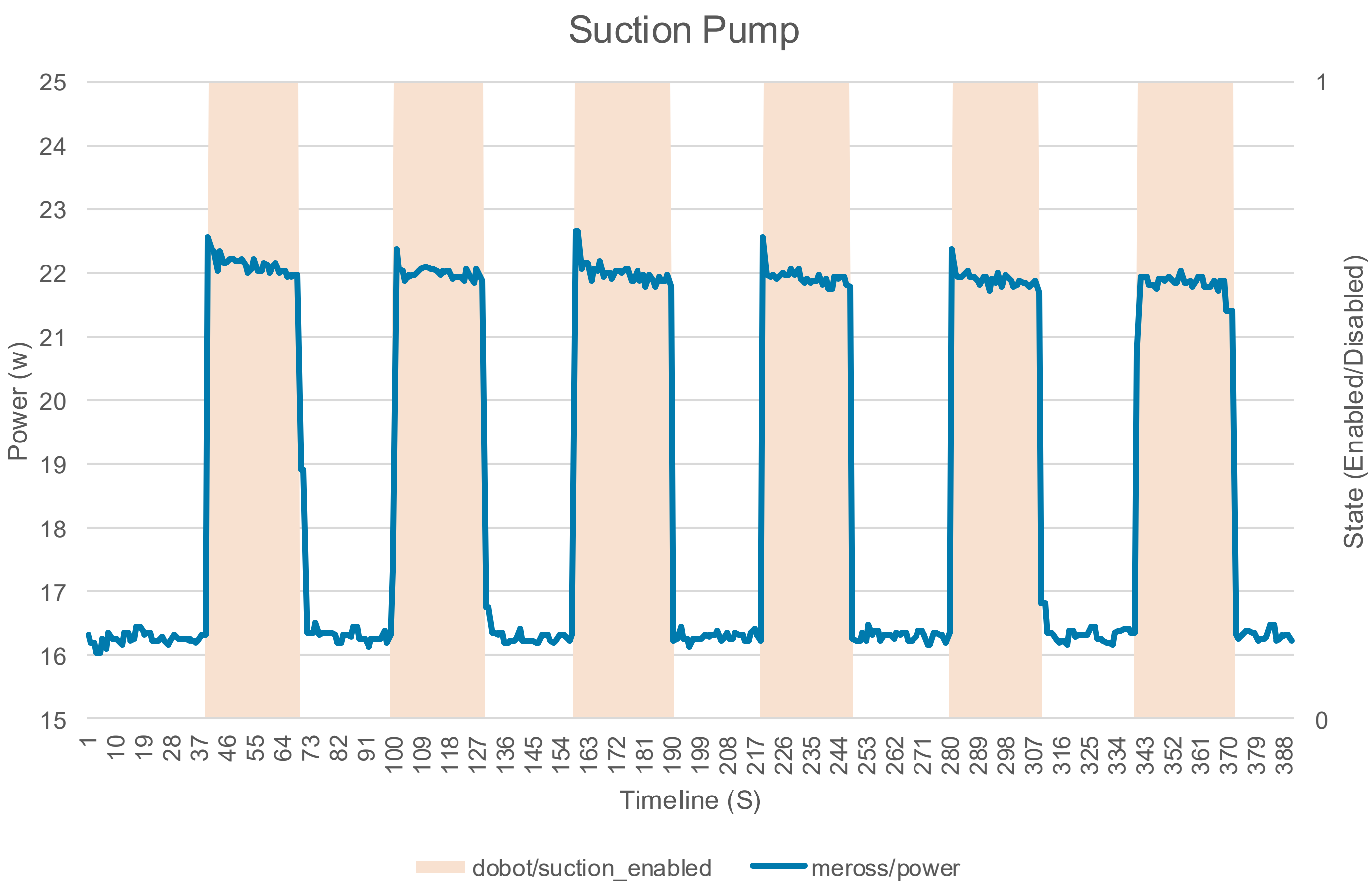


University of Cyprus
Department of Computer Science

Conveyor Belt

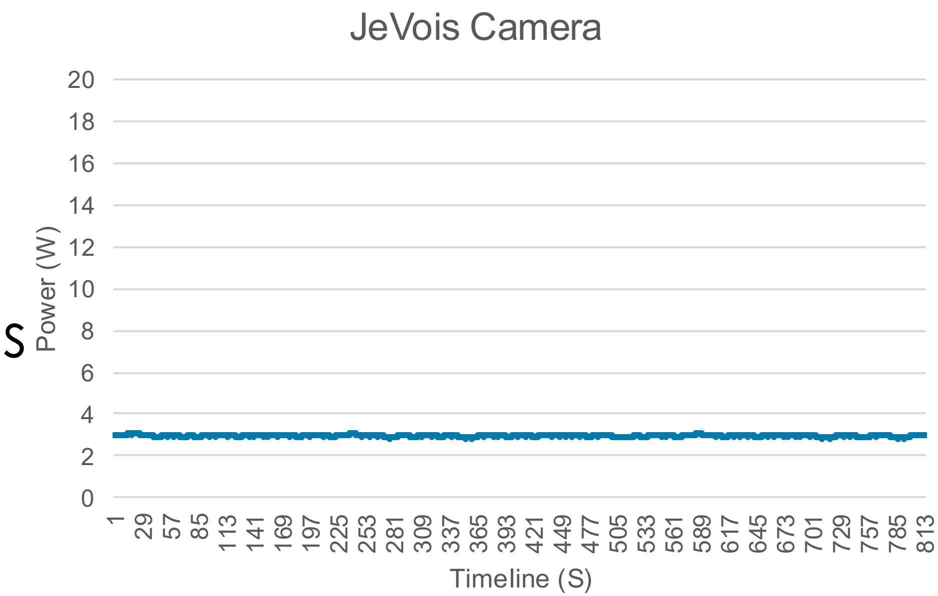


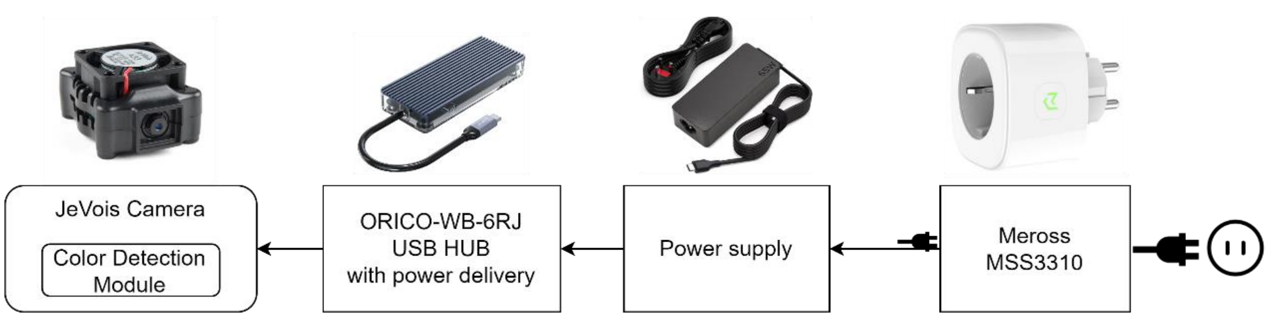
Suction Pump



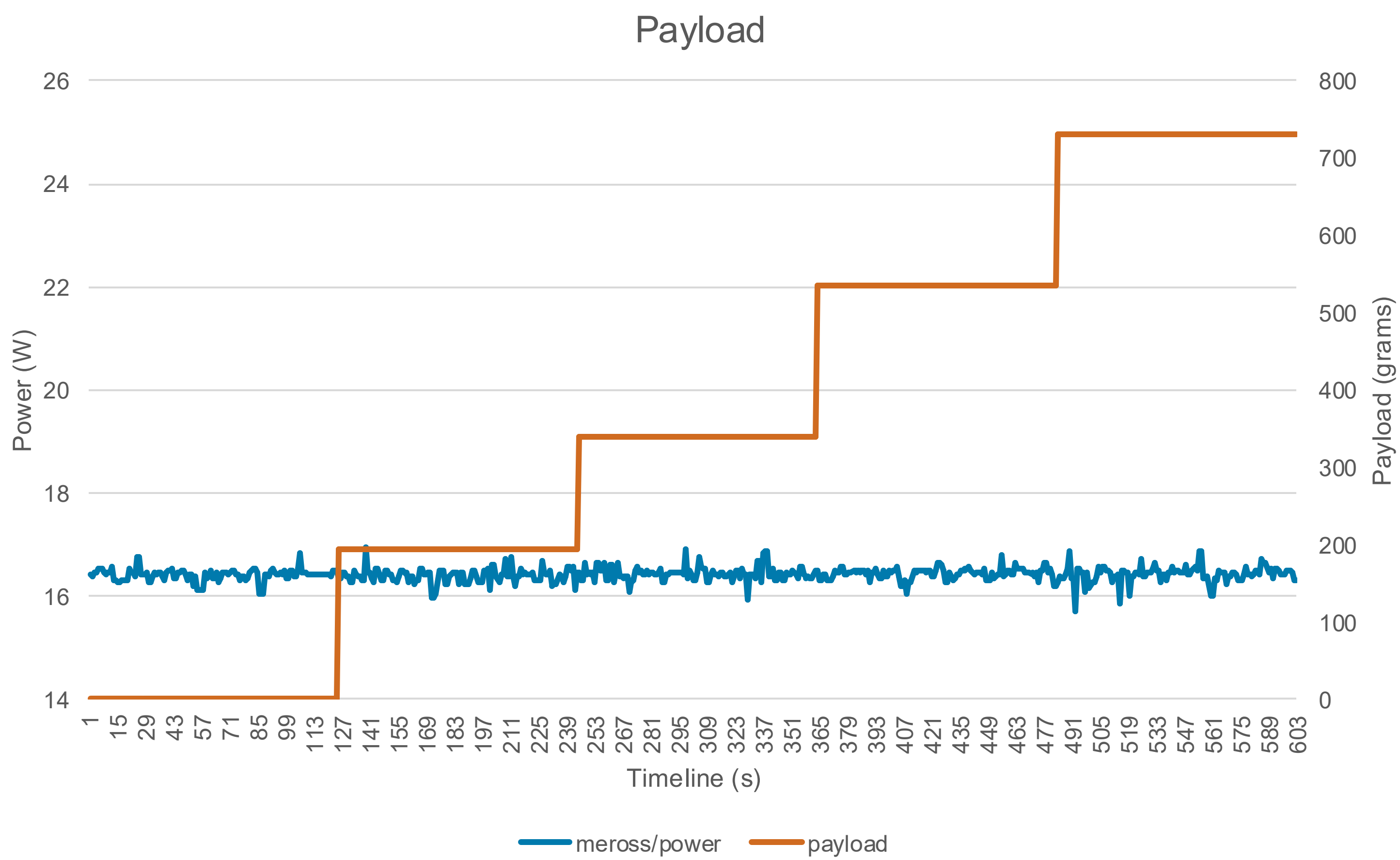
JeVois Camera

- Performing color detection
- $P_{cam} = P_{total} - P_{hub} \approx 2.13W$
- After 10 mins of execution
  - Power consumption remains stable
  - Excluded from the power consumption model

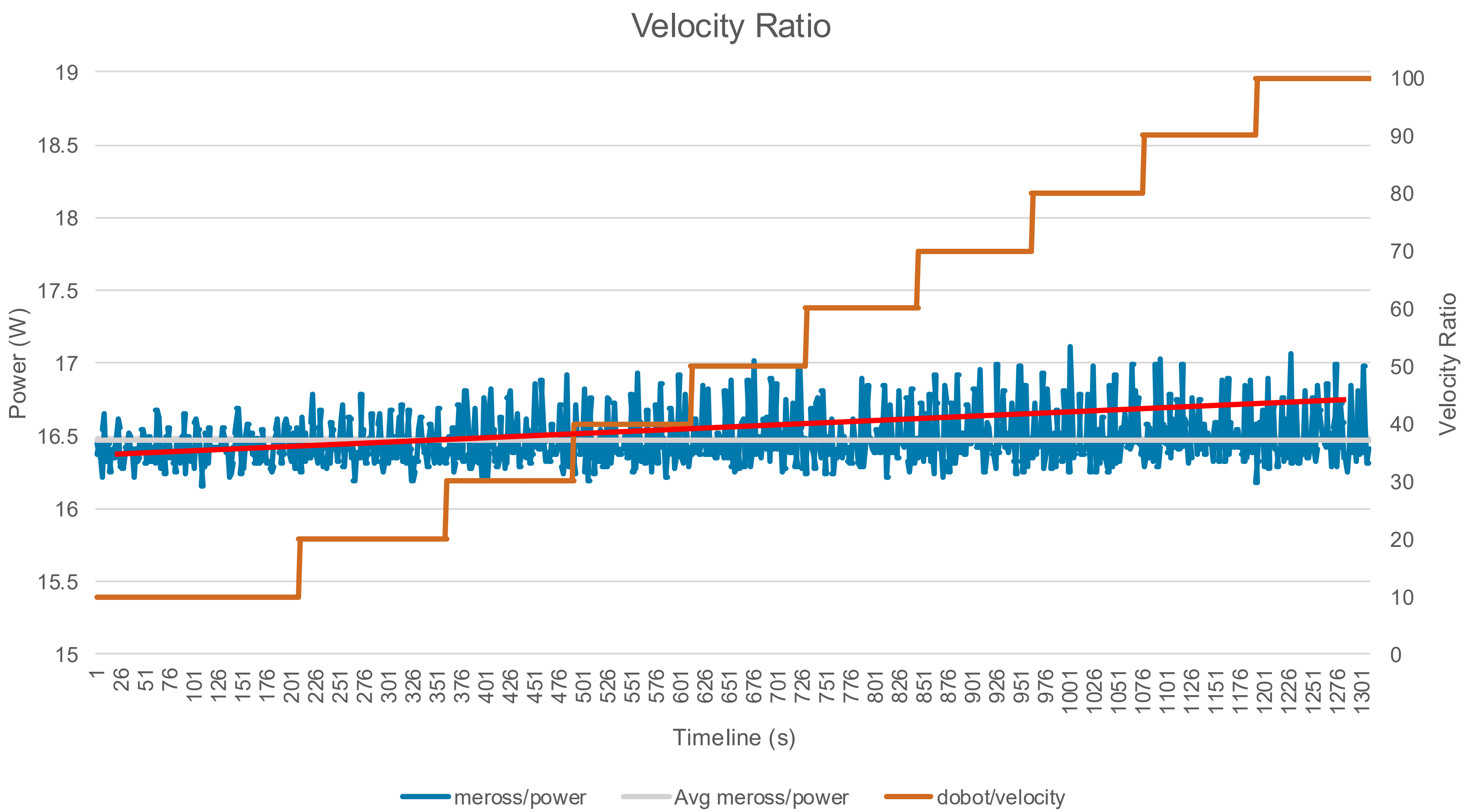




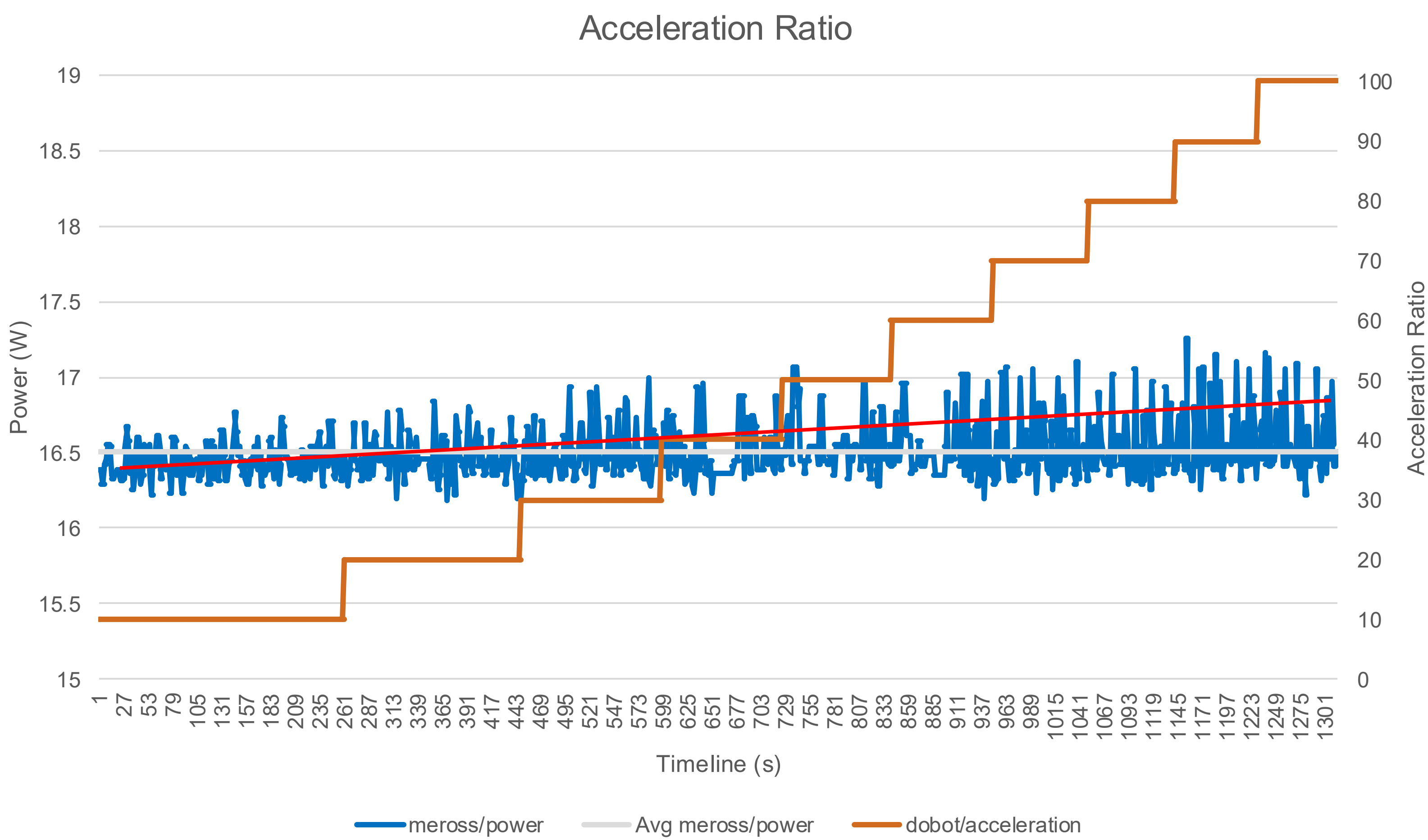
Robotic Arm - Payload



Robotic Arm: Velocity



Robotic Arm: Acceleration

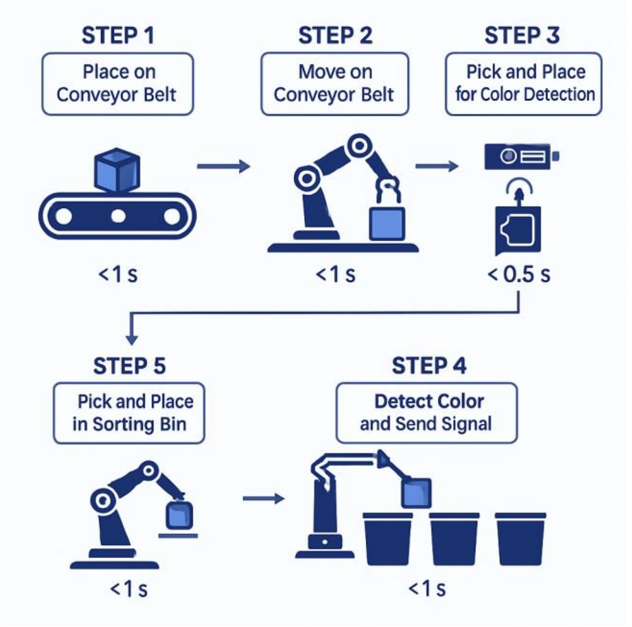


Key Takeaways

- **Smart camera** has stable power consumption when performing color inference
- **Arm velocity**, **acceleration**, and **payload** parameters have minimal influence on power consumption
- The **suction end-effector** is the component with the highest power consumption when enabled (performing suction)
- The **belt speed** influences the power consumption in an unexpected way, most probably due to physical interaction between the belt's components.

Workloads: End-to-end Application

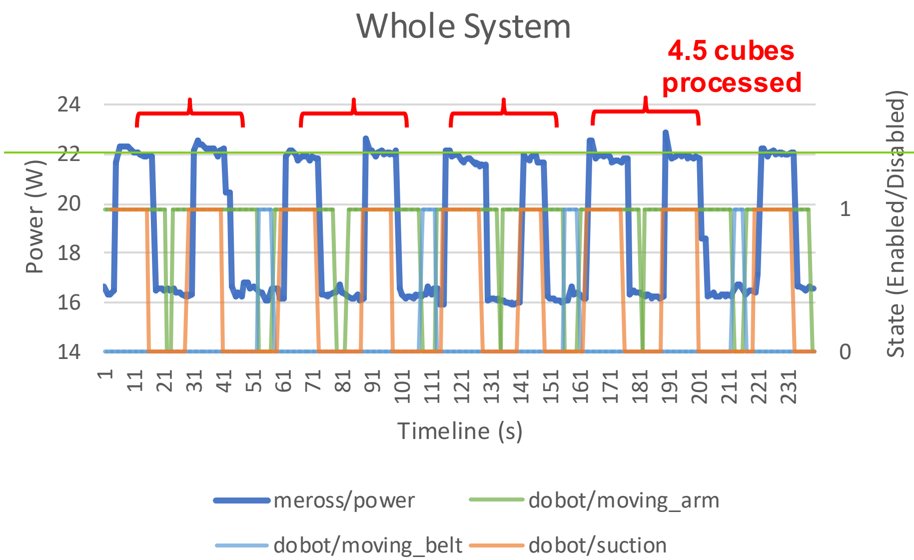
- Aim: simulate a typical industrial sorting process
  - Step 1: Place colored cube on conveyor belt.
  - Step 2: Belt moves cube near robotic arm.
  - Step 3: Arm picks cube and presents to smart camera.
  - Step 4: Camera detects color → controller updates.
  - Step 5: Arm places cube in designated box.
  - Repeat until no cubes remain.
- Configurable parameters:
  - Arm velocity: 30–100
  - Arm acceleration: 20–100
  - Belt speed: 10–80
  - Payload weight

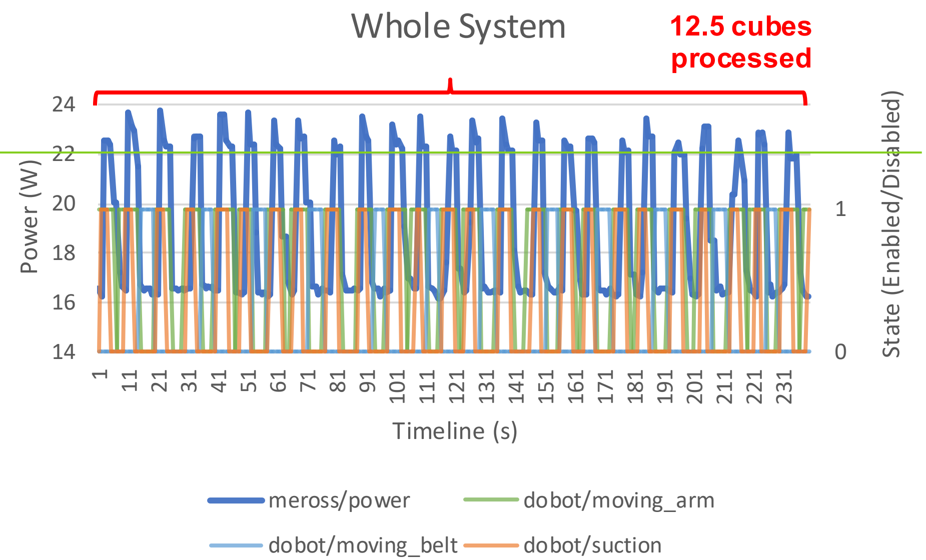


Dataset Contents

- acceleration_ratio (numerical 0-100)
- belt_speed (numerical 0-100)
- **operation1*** (categorical)
- colour (Red, Green, Blue)
- moving_belt (True/False)
- sample_timestamp (timestamp)
- current (Amperes)
- pose_x (numerical)
- pose_z (numerical)
- pose_joint1Angle (numerical)
- pose_joint3Angle (numerical)
- velocity_ratio (numerical 0-100)
- payload (grams)
- **operation2**** (categorical)
- moving_arm (True/False)
- suction (True/False)
- power (Watts)
- voltage (Volts)
- pose_y (numerical)
- pose_rHead (numerical)
- pose_joint2Angle (numerical)
- pose_joint4Angle (numerical)

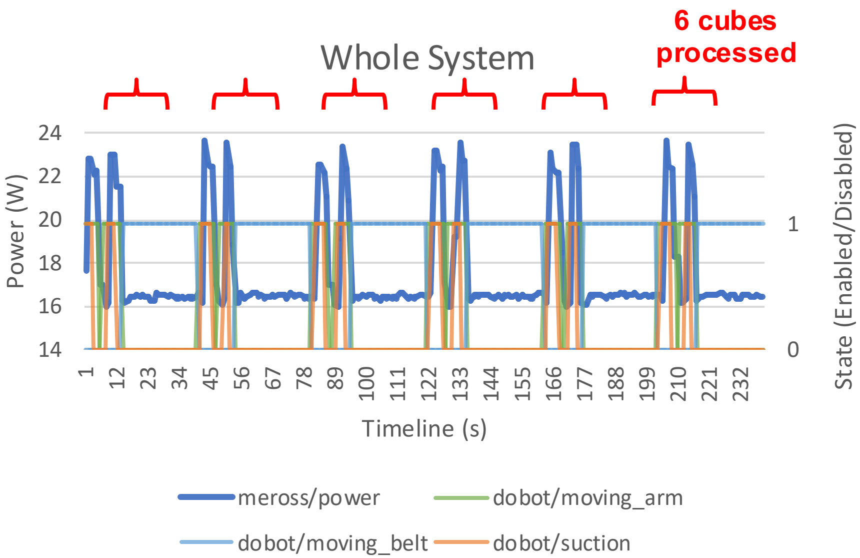
Experiments

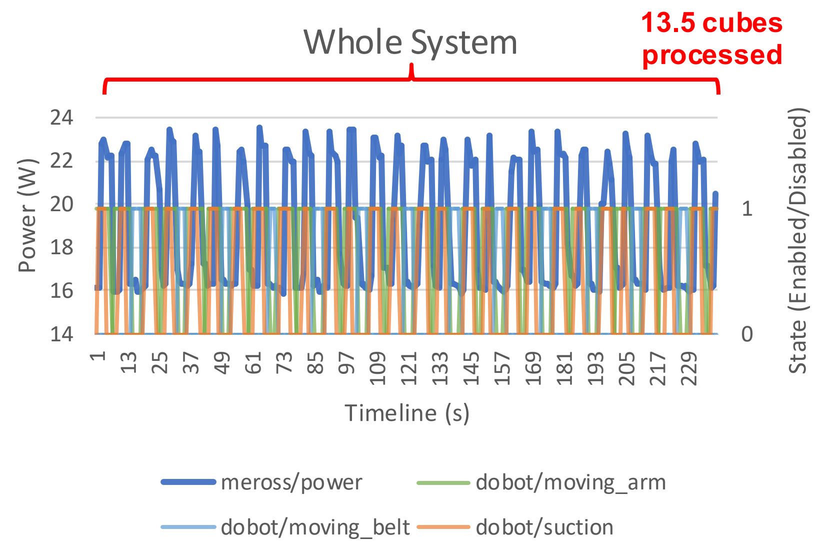




- Velocity ratio of **30** and **90**
- Acceleration ratio: **100**
- Conveyor belt speed: **60**
- Payload: **32** grams

Experiments





- Velocity ratio: **100**
- Acceleration ratio: **100**
- Conveyor belt speed: **10** and **70**
- Payload: **32** grams

Collected Dataset

- Data from the experiments are consolidated into a single dataset with 23,688 rows of data (≈ 6 hours and 35 minutes)
- Omitted data not useful for the training process:
  - Homing operations performed between experiments
  - Empty entries during the calibration Etc.
- One-Hot Encoding to convert categorical features

Dataset Preprocessing

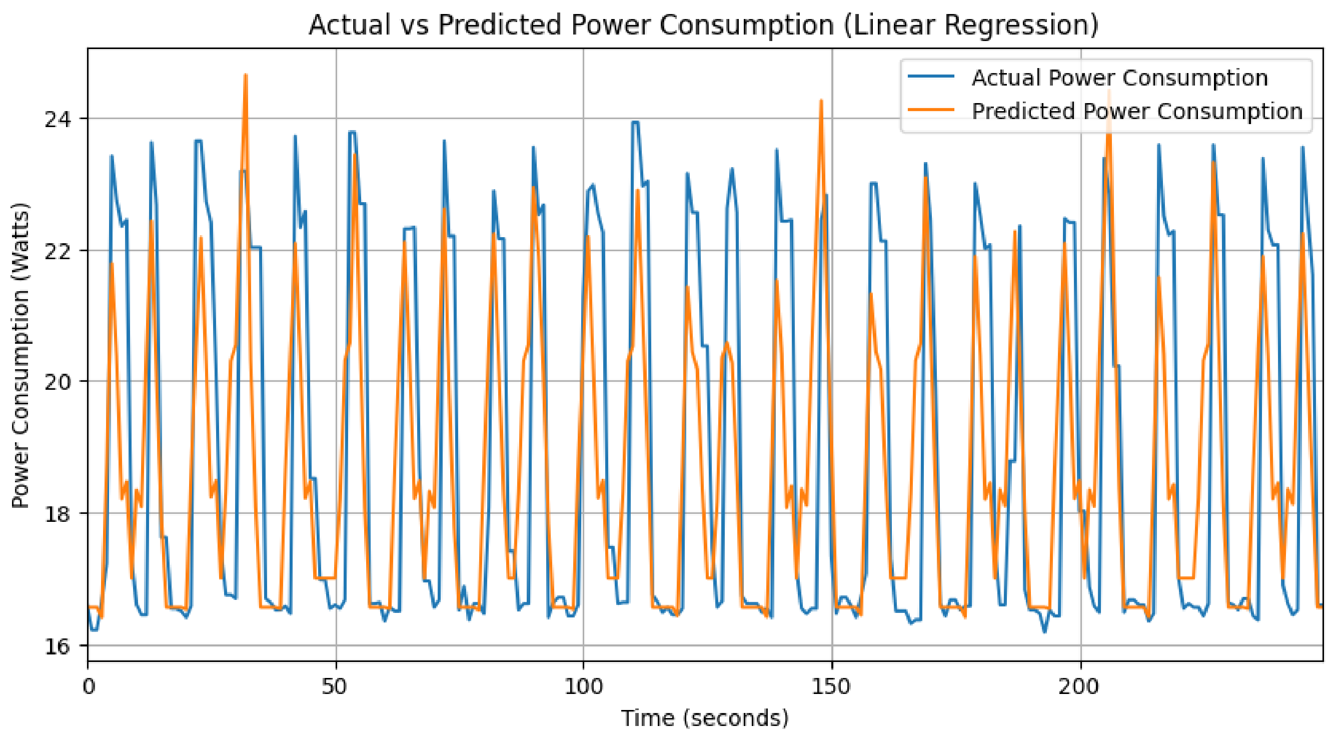
- Dataset split: Training 80% - Testing 20%
- Feature Selection:
  - Correlation Coefficient Filtering
    - Between the input features and power
  - Random Forest Importances
    - Train a random forest model, export final importances
  - Domain Knowledge
    - From the prior analysis
  - Empirical Analysis
    - Try different combinations

Model Training & Evaluation

- Linear Regression
- Polynomial Regression
- Decision trees
- Random Forest
- Gradient Boosting
- Support Vector Regression (SVR)
- Multi-Layer Perceptron (MLP)
- **Metrics**: Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE), Coefficient of Determination (R^2), Accuracy

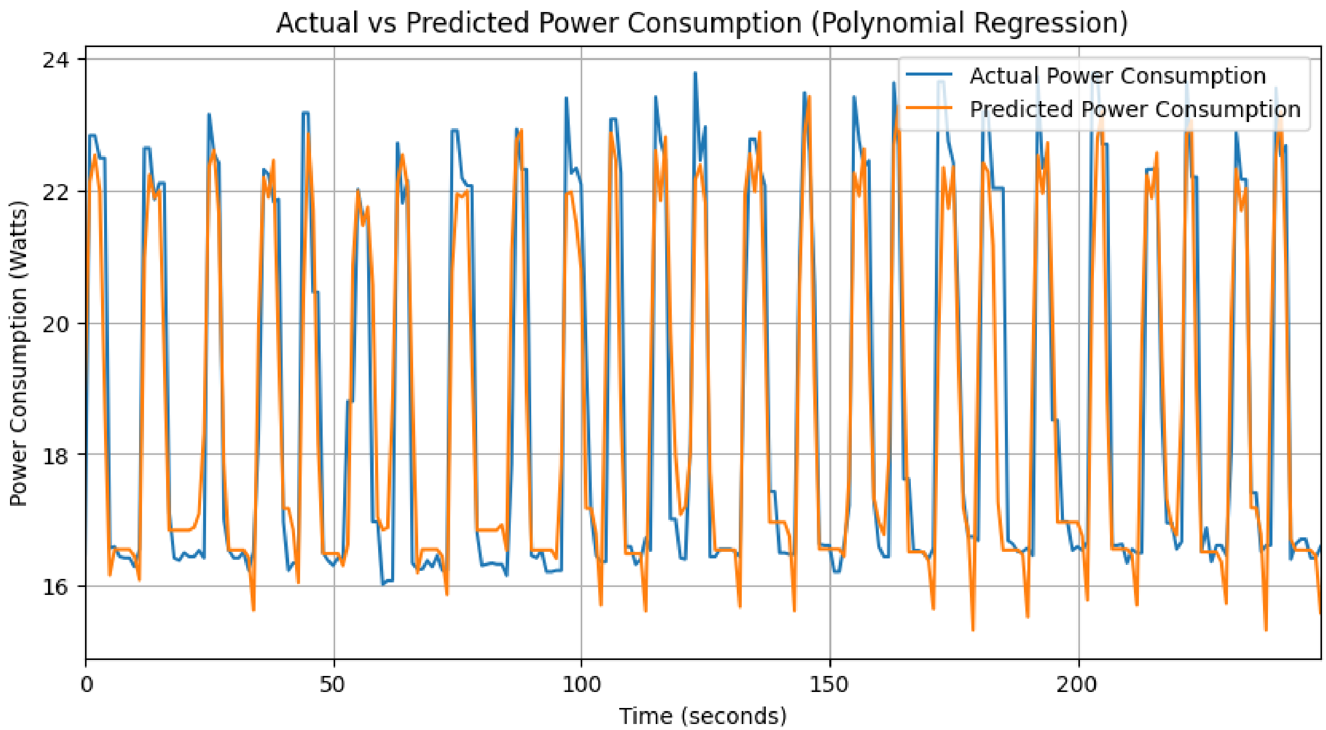
Linear Regression

- MAPE: 7.675%
- MAE: 1.463w
- R^2: 0.099
- Accuracy: 92.32%



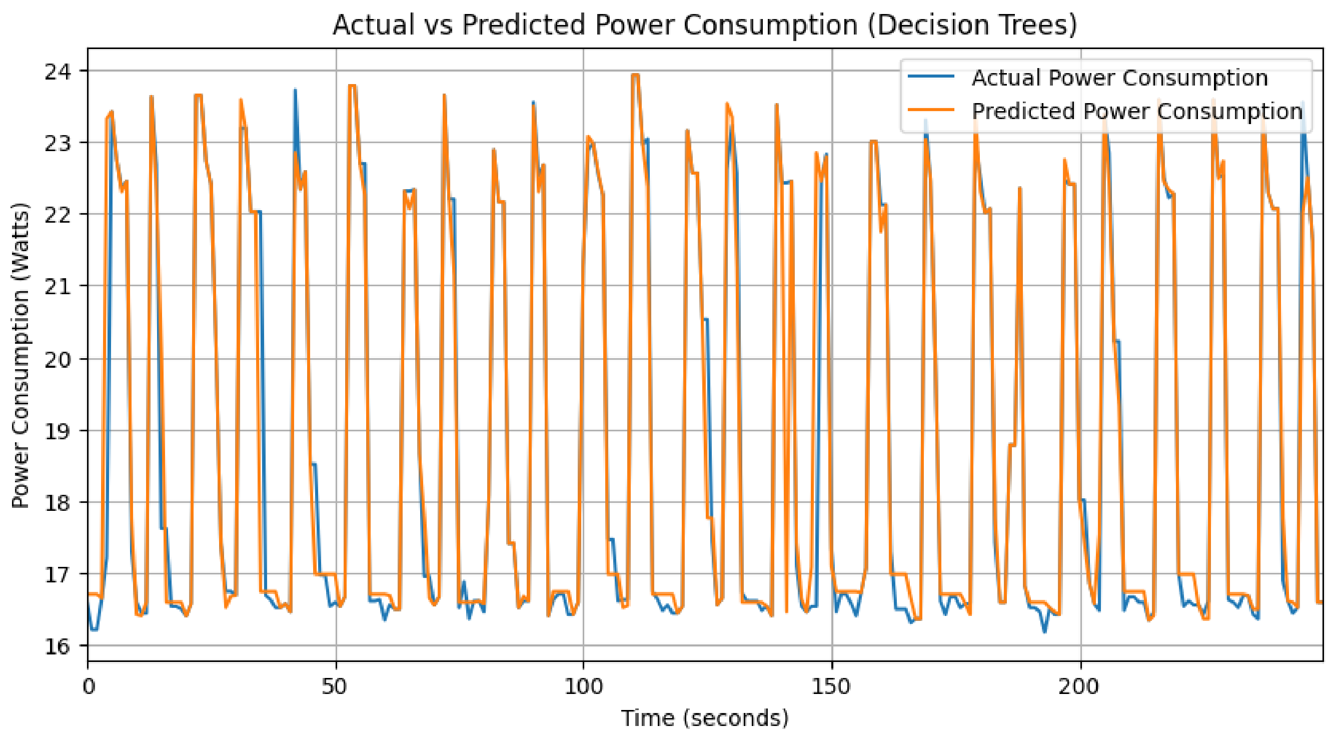
Polynomial Regression

- MAPE: 3.42%
- MAE: 0.65 w
- R^2: 0.82
- Accuracy: 96.58%



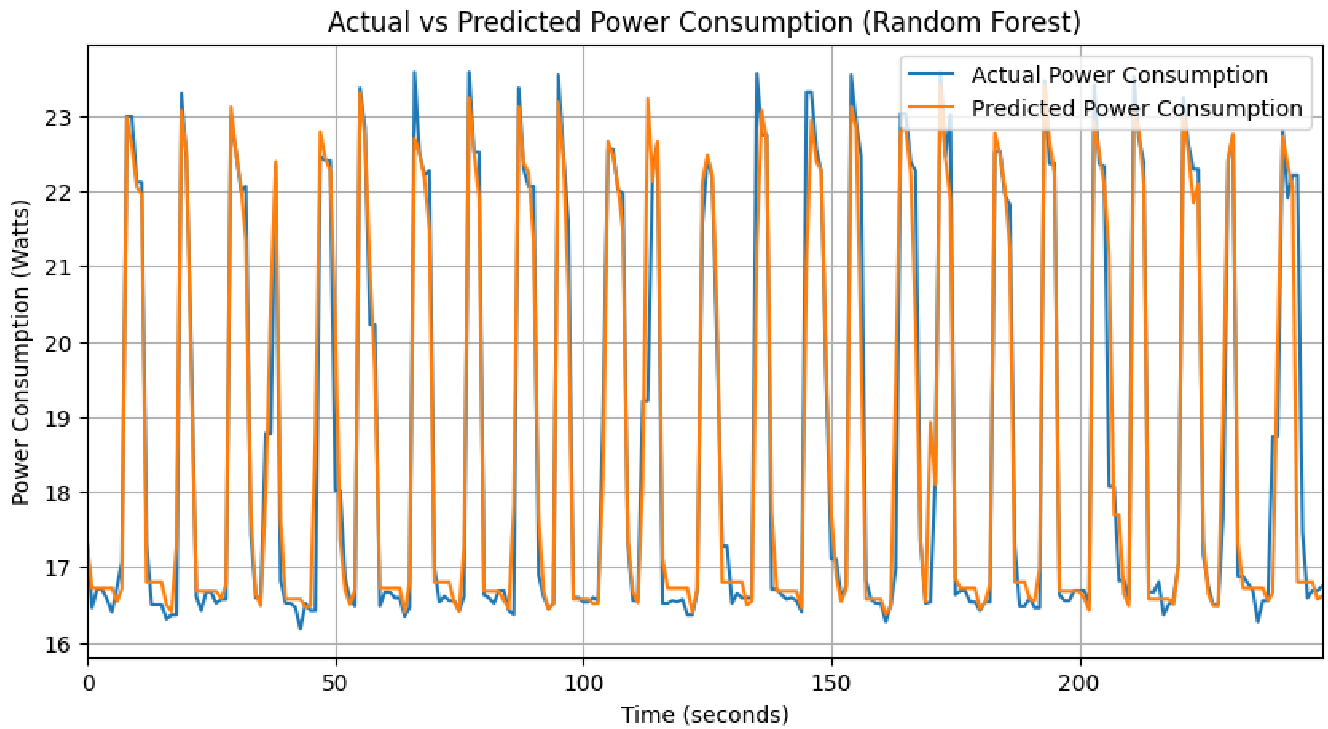
Decision Trees

- MAPE: 2.97%
- MAE: 0.57 w
- R^2: 0.77
- Accuracy: 97.03%



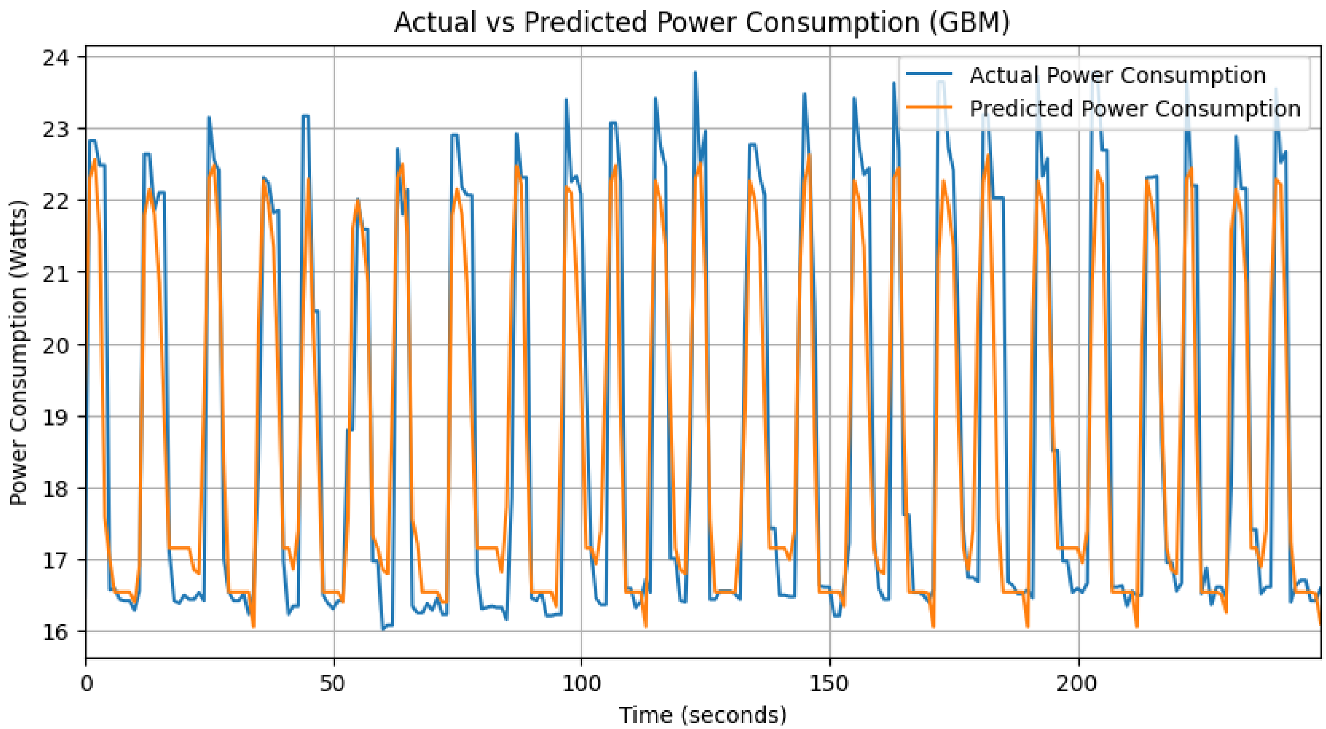
Random Forest

- MAPE: 2.6%
- MAE: 0.5 w
- R^2: 0.84
- Accuracy: 97.4%



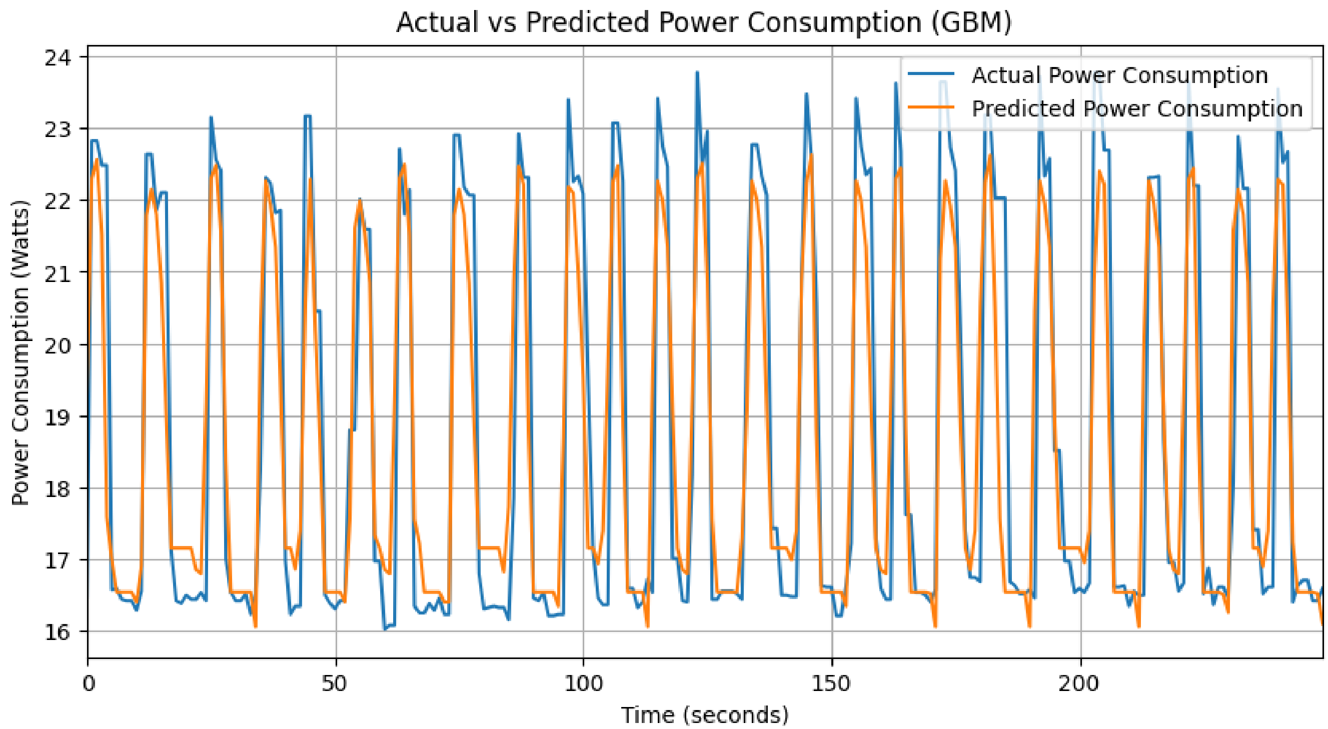
Gradient Boosting

- MAPE: 4.21%
- MAE: 0.8 w
- R^2: 0.72
- Accuracy: 95.78%



Support Vector Regression

- MAPE: 9.68%
- MAE: 1.79 w
- R^2: -0.44
- Accuracy: 90.32%



Summary

Model	MAPE Mean Absolute Percentage Error	MAE Mean Absolute Error	R^2 Coefficient of Determination	Accuracy
Linear Regression	7.675484823618989%	1.46	0.09987511216827072	92.32 %
Polynomial Regression	3.4184752310342246%	0.65	0.8190193031242793	96.58 %
Decision Tree	2.9676378452412617%	0.57	0.767490514574384	97.03 %
Random Forest	2.604260887103133%	0.5	0.8437542980071701	97.4 %
Gradient Boosting	4.216617347090262%	0.8	0.7169168078946087	95.78 %
Support Vector Regression (SVR)	9.677505508697633%	1.79	-0.4389872201446352	90.32 %
Multi-Layer Perceptron (MLP)	3.64785229769228%	0.69	0.824391507520702	96.35 %.

Summary

- **Accuracy** 90% - 97%
- Mean Absolute Percentage Error (**MAPE**): 2.6% - 9.68%
- Mean Absolute Error (**MAE**): 0.5 watts - 1.79 watts
- Best Performing:
  - **Random Forest Model**
  - Decision Tree Model
  - Polynomial Regression Model
- Worst Performing
  - **Support Vector Regression** (SVR)

Conclusions

- Methodology generated accurate models for round energy and duration
- Best model (**Random Forest**) has an error up to 4.23% in both target metrics, with other models providing comparable results.
- **Most important features** for both energy and round duration:
  - robotic arm acceleration followed by
  - arm velocity and
  - belt speed.

Future Work

- Collect additional features (e.g. camera temperature)
- Experiment with more application workloads
- Add and experiment with different end-effectors and robotic arms

http://linc.ucy.ac.cy

https://youtu.be/0hC_Sor5kEs?si=YOSnkVHRdd-CXwM0