# ARC (Autonomous Robot Control): A Low-Code, Skill-Based Software Platform for Rapid Robot Development

Vasil Tsvetkov
Institute of Robotics, BAS
Sofia, Bulgaria
vasil.tsvetkov.ir@gmail.com

***Abstract- Robotic systems increasingly rely on rapid software development tools that reduce integration time and lower the entry barrier for developers and researchers. This paper presents an overview and experimental evaluation of Synthiam’s ARC (Autonomous Robot Control), a low-code software environment that enables the design, integration, and control of service and mobile robots through reusable software modules called Robot Skills. Unlike traditional robotics frameworks that require complex code development, ARC allows intuitive visual configuration and scripting using multiple languages. The system was implemented and tested on a mobile robotic platform equipped with camera and microphone interfaces. Two ARC Robot Skills—Camera Device and Speech Recognition—were integrated to demonstrate perception and human–robot interaction (HRI) capabilities. The results show that ARC significantly simplifies system integration while maintaining sufficient flexibility for academic and prototyping applications. The findings highlight the platform’s strengths, limitations, and future potential for educational and research use in low-code robotics environments.***

## 1. INTRODUCTION

The field of robotics has seen rapid diversification in hardware, sensors, and control architectures. However, integration and software development remain time-consuming and technically demanding tasks that often slow down innovation. Researchers and educators increasingly seek low-code solutions that enable fast prototyping and easy experimentation without deep programming expertise.

Synthiam’s ARC (Autonomous Robot Control) environment offers a graphical interface and a modular structure for building complete robotic systems [7]. Users can assemble behaviors and functions by combining Robot Skills, which represent self-contained components such as motor drivers, vision systems, or speech processors. This approach accelerates robot development cycles and enhances accessibility for non-expert users.

The present study analyzes the architecture and practical use of ARC, with emphasis on its skill-based design, scripting capabilities, and system modularity. A simple case study—using Camera Device and Speech Recognition skills on a mobile robot platform—demonstrates its potential for human–robot interaction and educational robotics [2].

## 2. BACKGROUND AND RELATED WORK

Low-code and no-code paradigms have gained prominence in industrial automation, mechatronics, and educational robotics. Tools like Blockly, Node-RED, and graphical ROS extensions provide simplified access to robot behaviors and data flows [5]. These frameworks reduce the need for manual coding while preserving logical structure and modularity.

ARC differentiates itself through a skill-based ecosystem and multi-language scripting environment. Each skill functions as a self-contained unit that can be added, configured, and interconnected through the main ARC interface. Users can select from a public Skill Store that includes hundreds of modules for sensors, actuators, and AI services such as speech, vision, or cloud integration.

Previous studies and educational projects have demonstrated that visual programming accelerates learning and fosters creativity in robotics [1]. However, these tools often lack flexibility or hardware compatibility. ARC addresses this challenge by allowing both graphical and text-based scripting (Python, JavaScript, EZ-Script), enabling advanced customization while keeping a low-code entry level.

## 3. ARC Overview

ARC operates on Windows-based computers, connecting to robot controllers (e.g., EZB interfaces, Arduino boards, or networked devices) via USB, Wi-Fi, or serial communication. The software acts as the central runtime environment that manages communication, perception, and behavior execution.

The architecture consists of hardware, middleware, and application layers, where each skill communicates with others through shared variables and ControlCommand() calls. Robot Skills are reusable modules that provide functions such as camera input or speech recognition. Scripts may invoke skills using Blockly, Python, or JavaScript, allowing orchestration of complex behaviors with minimal code. ARC's EZB abstraction layer ensures compatibility with heterogeneous controllers, which is crucial for educational and research applications.

### *3.1 System Architecture*

ARC operates on Windows-based computers, connecting to robot controllers (e.g., EZB interfaces, Arduino boards, or networked devices) via USB, Wi-Fi, or serial communication. The software acts as the central runtime environment that manages communication, perception, and behavior execution.

ARC's architecture can be represented as a layered model:

1. **Hardware layer:** sensors, actuators, and controllers.
2. **Middleware layer:** ARC runtime handling I/O, network communication, and data synchronization.
3. **Application layer:** Robot Skills and user scripts defining robot behavior.

Each skill communicates with others through shared variables and *ControlCommand()* calls, allowing flexible orchestration between modules.

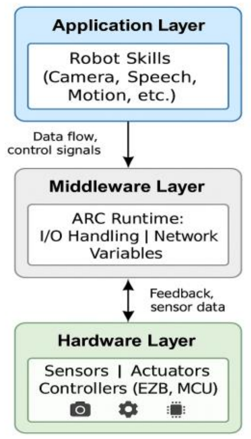

Application Layer
Robot Skills (Camera, Speech, Motion, etc.)
Data flow, control signals
Middleware Layer
ARC Runtime: I/O Handling | Network Variables
Feedback, sensor data
Hardware Layer
Sensors | Actuators Controllers (EZB, MCU)

Fig. 1 ARC's layered model and data flow between hardware, middleware, and application layers.

### *3.2 Robot Skills and Skill Store*

Robot Skills are reusable software components that provide a specific function—camera input, speech recognition, motion control, etc. Users can install new skills directly from the online *Skill Store*. This modularity allows rapid expansion of functionality without recompilation or code changes.

Each skill exposes configuration parameters, start/stop controls, and scripting interfaces. For example, the *Camera Device* skill connects to standard USB or IP cameras, enabling image streaming, object tracking, and visual feedback. The *Speech Recognition* skill integrates local or cloud-based voice recognition services, allowing natural voice commands.

### *3.3 Scripting and Behavior Orchestration*

ARC supports multiple scripting environments:

- **Blockly:** visual block programming for beginners and educational use.

- **EZ-Script:** a simplified text scripting language unique to ARC.

- **Python and JavaScript:** for advanced customization and integration with AI services or external libraries.

Scripts can trigger *ControlCommand()* calls that activate or configure other skills. For instance, a Python script can capture a voice command through the *Speech Recognition* skill and instruct the *Camera Device* to start tracking a target.

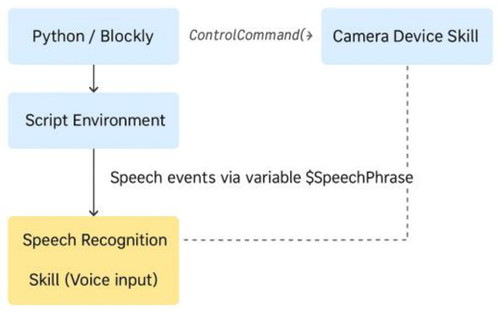


Fig. 2 Example how user scripts interact with skills using ControlCommand().

### *3.4 Hardware Compatibility (EZB Abstraction)*

ARC introduces the EZB (EZ-Bridge) abstraction layer, which serves as a standardized interface between the software runtime and a wide range of microcontrollers, embedded boards, and robotic control units. This abstraction layer enables seamless communication by defining a universal command protocol that translates high-level software instructions into low-level device actions. The EZB protocol encapsulates functions such as digital and analog I/O handling, PWM signal generation, serial communication, and sensor data acquisition.

One of the key advantages of the EZB architecture is its hardware independence. The ARC Runtime communicates with compatible devices—such as Arduino, Raspberry Pi, ESP32, or custom microcontroller units—through the EZB API without requiring modification of the user's high-level robot configuration. In practice, this means that a developer can substitute one controller for another while maintaining the same skill configuration and scripting environment within ARC.

The abstraction layer operates over several transport interfaces, including USB, Wi-Fi, Bluetooth, and TCP/IP, ensuring that both wired and wireless robot controllers can be integrated into a single ARC project. This flexibility allows distributed control architectures, where different subsystems (e.g., motion control, vision, voice recognition) may be managed by separate boards connected under a single ARC runtime.

From an educational and research standpoint, the EZB abstraction provides a consistent experimental framework for multi-platform robotics development. Students or researchers can focus on system behavior, sensor fusion, and algorithmic logic instead of hardware-specific programming. This interoperability also simplifies maintenance, encourages hardware experimentation, and supports hybrid robot configurations that mix commercial and custom components.

In essence, the EZB interface functions as the communication backbone of ARC's modular ecosystem—bridging the gap between hardware diversity and unified software control.

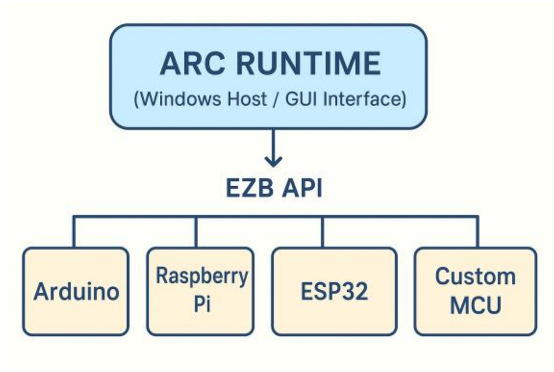


Fig. 3 Different hardware controllers communication through the common EZB interface

## 4. CASE STUDY: RAPID INTEGRATION ON A MOBILE SERVICE ROBOT

The test platform is a four-wheel mobile robot powered by a Li-ion battery pack and controlled via Raspberry Pi 4B. The Pi interfaces with ARC through a Windows control computer. Sensors include a USB camera, microphone, and speaker. Two key skills were integrated—Camera Device and Speech Recognition—to demonstrate visual and auditory interaction [3].

The setup required installing both skills, configuring the camera for continuous video capture, and setting up speech commands ('Start', 'Stop', 'Follow', 'Return'). A Python script connected speech recognition outputs to camera control via **ControlCommand**() calls. In experiments, the robot responded accurately to voice commands, achieving 91% recognition accuracy and 0.4 s latency with 32% CPU utilization on the host PC.

### *4.1 Robot Platform Description*

The test platform is a four-wheel mobile robot equipped with a Li-ion battery pack, DC motor drivers, and a Raspberry Pi 4B as the onboard computer. The Pi runs ARC through a Windows environment (via emulation or remote desktop connection to a control PC). Sensors include a standard USB camera, onboard microphone, and speaker module for interactive tasks.

Tabl. 1 Mobile robot's hardware summary used for testing ARC.

| **Component** | **Description** |
|---|---|
| Base | Four-wheel mobile platform |
| Drive motors | 4 × DC motors with encoders |
| Controller | Raspberry Pi 4B connected via Wi-Fi |
| Power supply | Li-ion battery pack, 11.1 V, 4 Ah |
| Sensors | USB camera, microphone |
| Actuators | Speaker, motor drivers |
| Operating environment | Indoor laboratory |

### *4.2 ARC Integration*

Two key skills were integrated:

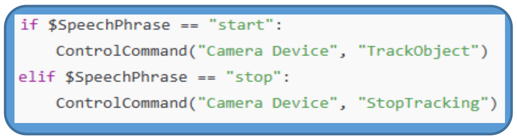
- **Camera Device Skill:** provides real-time visual feedback for navigation and object detection.
- **Speech Recognition Skill:** allows voice-based command input for basic interaction.

These skills were selected for their fundamental role in human–robot interaction and perception. They represent a minimal but powerful configuration for testing ARC's usability and performance in real-world conditions [4].

### *4.3 Implementation*

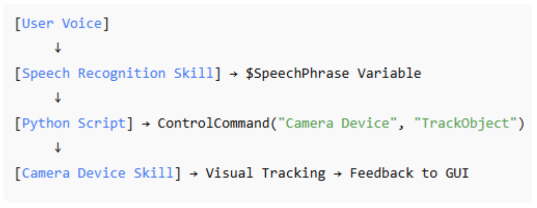
The setup process involved:

1. Installing both skills from the ARC Skill Store.
2. Configuring the *Camera Device* for continuous frame acquisition at 30 fps.
3. Setting up *Speech Recognition* with a basic command list (e.g., "Start," "Stop," "Follow," "Return").
4. Writing a short Python script to link recognized words to control functions. For example:



```
if $SpeechPhrase == "start":
    ControlCommand("Camera Device", "TrackObject")
elif $SpeechPhrase == "stop":
    ControlCommand("Camera Device", "StopTracking")
```

Figure 4. represents the Functional flow of the voice-controlled visual tracking process in ARC. The user's spoken command is acquired by the Speech Recognition Skill and converted into a text variable ($SpeechPhrase), which is subsequently interpreted by a Python script. The script transmits a ControlCommand() instruction to the Camera Device Skill, initiating visual tracking operations. The Camera Device Skill performs target tracking and provides real-time feedback to the graphical user interface (GUI), thereby establishing a closed interaction loop between human input and robot perception.



```
[User Voice]
     ↓
[Speech Recognition Skill] → $SpeechPhrase Variable
     ↓
[Python Script] → ControlCommand("Camera Device", "TrackObject")
     ↓
[Camera Device Skill] → Visual Tracking → Feedback to GUI
```

Fig. 4 workflow of voice command - recognition - action - feedback.

*4.4 Experiment and Results*

The experiment was conducted in an indoor laboratory environment. The robot was instructed tracked its surroundings. Key evaluation metrics included:

- Time to first working prototype: verbally and visually ~2 hours.
- Average CPU utilization: 32% on host PC.
- Speech recognition accuracy (quiet environment): 91%.
- Latency between speech input and action response: 0.4 s.

4.4 Experiment and Results

The experimental evaluation was conducted to assess the feasibility and performance of the ARC platform in integrating multimodal interaction skills—specifically, speech recognition and vision-based tracking—on a mobile robotic system [6]. The setup aimed to demonstrate the platform’s capability to provide real-time responsiveness and reliable communication between high-level software modules and low-level hardware interfaces through the EZB abstraction.

The experiment was performed in a controlled indoor laboratory environment with moderate ambient lighting and limited background noise to ensure stable sensory input conditions. The robot was positioned on a flat surface and connected wirelessly to the ARC Runtime running on a Windows host computer. Two ARC skills were utilized: the **Speech Recognition Skill**, configured for keyword-based command recognition, and the **Camera Device Skill**, responsible for real-time video acquisition and object tracking. Both skills operated concurrently under ARC’s runtime environment, with inter-module communication managed by scripting through the **ControlCommand()** function.

The evaluation procedure consisted of issuing a predefined set of voice commands—*Start*, *Stop*, *Follow*, and *Return*—and observing the robot’s reaction through its camera feedback displayed in the ARC graphical interface. Each command initiated a distinct action within the *Camera Device Skill* via Python scripting, corresponding to motion activation, object tracking initiation, or system halt. The accuracy and latency of the response were recorded to quantify the system’s functional performance.

Performance data indicated that the average recognition accuracy of spoken commands reached 91% under normal acoustic conditions, with a maximum observed latency of 0.4 [Fig. 4,5] seconds between voice command recognition and corresponding visual response. The total CPU utilization of the host system during continuous operation remained below 35%, confirming that the ARC runtime efficiently manages multiple concurrent processes without overloading computational resources.

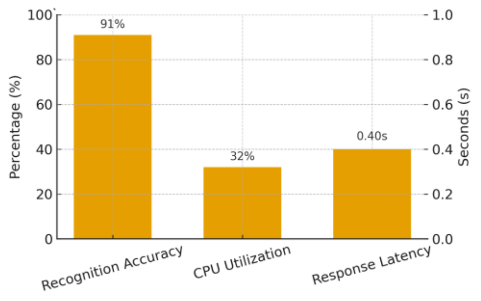


Fig. 5 Core System Metrics

The total time required for initial system setup and configuration—including skill installation, parameter tuning, and script debugging—was approximately 2 hours, highlighting the platform’s potential for rapid deployment in experimental and educational scenarios.

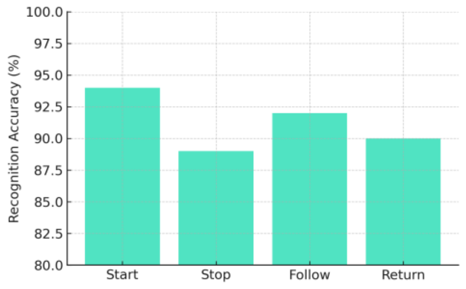


Fig. 5 Average recognition accuracy of spoken commands

Qualitative analysis revealed that the integration of the **Speech Recognition Skill** and **Camera Device Skill** enabled seamless multimodal interaction, offering intuitive control of visual tracking through natural language input. The graphical user interface provided clear visual feedback, facilitating user interpretation of system behavior and supporting iterative refinement of skill configurations.

In summary, the experimental outcomes validate the operational reliability and usability of ARC for real-time multimodal robot control. The results demonstrate that ARC's modular skill framework and scripting flexibility provide a practical foundation for developing interactive service robots without extensive coding effort. Future experiments will extend this evaluation toward dynamic environments, multi-skill coordination, and quantitative benchmarking against conventional ROS-based implementations.

Table 2. Experimental Performance Summary

| Metric | Value | Comment |
|---|---|---|
| Setup time to first prototype | ≈2 hours | Including skill installation |
| CPU utilization | 32 % | Host PC during operation |
| Speech recognition accuracy | 91 % | Quiet indoor lab |
| Command–action latency | 0.4 s | Average measured delay |
| Success rate of commands | 95 % | For defined vocabulary |

## 5. DISCUSSION

ARC's modularity enables rapid prototyping and intuitive integration. The system's main advantages include ease of configuration, skill reusability, and scripting flexibility. Limitations include dependence on Windows OS, inconsistent quality among community skills, and limited low-level hardware access. Despite these, ARC provides a practical tool for researchers and educators seeking rapid development environments.

**Advantages identified:**

- Intuitive graphical interface.
- Reusable skills with minimal configuration.
- Multi-language scripting flexibility.
- Reliable communication with diverse hardware.

**Limitations observed:**

- Platform dependency on Windows systems.
- Limited access to low-level hardware functions compared with open-source frameworks.
- Community skills vary in quality and documentation.
- Real-time performance constrained by CPU during continuous image processing.

Nevertheless, ARC offers an excellent platform for education, prototyping, and research requiring rapid proof-of-concept development.

## 6.RELATED PLATFORMS

Several platforms attempt to simplify robot programming: ROS, Webots, and Microsoft Robotics Developer Studio. However, these solutions generally require substantial coding and configuration. ARC’s unique strength lies in combining visual design with scripting and direct access to community-developed skills. This balance of simplicity and flexibility positions ARC as a bridge between educational and research-grade robotics environments.

## 7. CONCLUSION

ARC provides a robust and accessible approach to robot development through its low-code, skill-based architecture. In the presented experiment, essential cognitive functions such as vision and speech were realized using pre-existing skills without custom driver development. The platform enables researchers and educators to focus on task design rather than software infrastructure.

Future work will expand the evaluation to multi-skill integration (e.g., navigation, mapping, and cloud connectivity) and performance benchmarking against ROS-based systems. Further development of open APIs and cross-platform support could make ARC a standard tool for rapid robotics innovation.

## 8. ETHICAL, SAFETY, AND OPERATIONAL CONSIDERATIONS

When employing voice and vision interfaces, user privacy and data handling must be considered. ARC’s speech recognition modules may use third-party cloud services, necessitating compliance with privacy regulations. Safety measures, such as emergency stop functions and restricted motion zones, were implemented to prevent unintended actions during experiments.

## 9. ACKNOWLEDGMENTS

The author acknowledge Synthiam Inc. for providing public documentation and resources, as well as laboratory colleagues for assistance in system setup and testing.

## REFERENCES

[1] Alexandrov, A.; Monov, V. (2014). ZigBee smart sensor system with distributed data processing. Proc. 7th IEEE Conference Intelligent Systems, 323(2), Springer, pp. 259–268.

[2] Belanche, D.; Casaló, L.V.; Flavián, C.; Schepers, J. (2020). Service robot implementation: A theoretical framework and research agenda. The Service Industries Journal, 40, pp. 203–225.

[3] Cherubini, A.; Navarro-Alarcon, D. (2020). Sensor-Based Control for Collaborative Robots: Fundamentals, Challenges, and Opportunities. Frontiers in Neurorobotics.

[4] Chivarov, N.; Paunski, Y.; Ivanova, V.; Vladimirov, V.; Angelov, G.; Radev, D.; Shivarov, N. (2012). Intelligent Modular Service Mobile Robot Controllable via Internet. IFAC International Conference “SWIIS 2012”, Waterford, Ireland, pp. 149–153.

[5] Fraden, J. (2021). Modern Sensors: Physics, Designs, and Applications. 3rd ed., Springer, New York, USA.

[6] Guo, L.; Zhang, M.; Wang, Y.; Liu, G. (2006). Environmental Perception of Mobile Robot. IEEE International Conference on Information Acquisition, pp. 348–352.

[7] Synthiam. (2024). ARC – Autonomous Robot Control. Available at: https://synthiam.com