CUE CLEAR: A WEARABLE AUGMENTATIVE AND ALTERNATIVE COMMUNICATION SYSTEM TO DISPLAY CAREGIVER-CURATED SYMBOL SLIDESHOWS USING A FLUTTER MOBILE APPLICATION AND A WI-FI-CONNECTED CIRCUIT PYTHON EMBEDDED DEVICE

Yeuk Nam Nolan Liu[1] and Tyler Boulom[2]

[1]Lexington High school, 251 Waltham St, Lexington, MA 02421
[2]Woodbury University, 7500 N Glenoaks Blvd, Burbank, CA 91504

ABSTRACT

Many non-verbal individuals depend on symbol-based communication, yet AAC devices are frequently abandoned because they are bulky, conspicuous, or hard to reconfigure. Cue Clear addresses this with a wearable display that cycles through caregiver-curated communication symbols, controlled by a companion smartphone app. The system pairs a CircuitPython microcontroller driving an ILI9341 LCD—hosting a non-blocking HTTP server over a Wi-Fi access point—with a cross-platform Flutter application that builds symbol "profiles" and activates them on the device [3]. Key challenges included reliable communication with a resource-constrained device, responsive image browsing, and an accessible caregiver interface; these were met with a non-blocking embedded server, a read-through image cache, and a large-target, confirmation-guarded UI. In testing, caching cut image load latency about 27-fold to 43 ms, and index-only profile activation stayed near 215 ms regardless of profile size. By directly targeting known causes of AAC abandonment, Cue Clear offers a sustainable, low-friction communication aid.

KEYWORDS

Augmentative and Alternative Communication (AAC), Picture Exchange Communication System (PECS), CircuitPython, Flutter, SoftAP HTTP server

1. INTRODUCTION

Communication is a fundamental human right, yet a substantial population cannot rely on spoken language to meet it. An estimated 1.3% of the global population—roughly 97 million people—live with complex communication needs that benefit from augmentative and alternative communication (AAC) [1]. These individuals include people with autism spectrum disorder, cerebral palsy, aphasia following stroke, and amyotrophic lateral sclerosis. For non-verbal individuals, symbol-based systems such as the Picture Exchange Communication System (PECS) have demonstrated meaningful gains in functional requesting and expressive communication [4][5]. Meta-analyses confirm that aided AAC produces moderate-to-large improvements in communication outcomes for children with autism [6].

David C. Wyld et al. (Eds): IPPR, CSIT, NLPA, AMLA – 2026
pp. 01-24, 2026. CS & IT - CSCP 2026 DOI: 10.5121/csit.2026.1601402

Despite this evidence, AAC adoption remains fragile. Studies report that roughly one-third of AAC devices are abandoned, frequently because the technology is bulky, socially conspicuous, difficult for caregivers to reconfigure, or poorly matched to the user's daily routines [9][10]. The shift toward tablet-based AAC apps improved portability and reduced cost, but these solutions still demand that the user hold and operate a screen, and they do not always integrate naturally into a child's clothing or line of sight during everyday activities [11][12]. A persistent gap therefore exists: an AAC display that is genuinely wearable, glanceable, low-cost, and trivially reconfigurable by a non-technical caregiver. Closing this gap is important because reliable, low-friction access to communication symbols directly affects autonomy, social participation, and quality of life for some of the most vulnerable members of society [2].

Tablet-based AAC (McNaughton and Light) lowered cost and stigma and widened access, but it requires the user to actively hold and operate a multipurpose screen and offers no hands-free persistent display; Cue Clear adds a dedicated always-on wearable surface while keeping phone-based authoring.

The Picture Exchange Communication System (Bondy and Frost) is an evidence-based, low-cost symbol method, but it depends on printing, laminating, carrying, and manually reorganizing physical cards with no remote update path; Cue Clear keeps the symbol model while making reconfiguration instant and wireless.

The systematic review by Moorcroft and colleagues catalogued why AAC is abandoned—poor routine fit, maintenance burden, and weak customization—but proposed no concrete system; Cue Clear answers those specific barriers with a worn, printing-free device that a caregiver reconfigures in seconds from a phone they already own, aiming for more sustainable daily use.

This project proposes Cue Clear, a wearable AAC system in which a small Wi-Fi-connected display worn on a person's clothing cycles through a slideshow of communication symbols that a caregiver curates and pushes from a companion mobile application. The solution decouples the display—a lightweight, glanceable embedded device—from the authoring tool—a familiar smartphone app—so that each can be optimized for its role.

The hardware is a Circuit Python microcontroller driving an ILI9341 LCD that hosts its own Wi-Fi access point and a minimal HTTP server. The companion tool is a cross-platform Flutter application that connects to the device, lets a caregiver assemble named "profiles" (a curated set of symbols plus a display interval), and activates a profile on the device with a single tap. Because activation transmits only a compact list of image indices rather than image data, reconfiguration is near-instantaneous.

This design is effective because it attacks the documented causes of AAC abandonment directly: the device is unobtrusive and worn rather than held, the symbols are always visible without user operation, and the caregiver workflow is reduced to selecting pictures on a phone they already own. Compared with tablet-only AAC apps, Cue Clear adds an always-on wearable surface; compared with low-tech laminated card systems, it adds remote, instant, and unlimited reconfiguration [9][11]. The combination of a familiar mobile authoring experience with inexpensive embedded hardware offers a more sustainable path to daily AAC use.

Two experiments evaluated the system's responsiveness. The first targeted the image caching layer, asking whether it truly removes redundant downloads. The same symbol was loaded eight times both as a fresh fetch over the device's access point and as a cached retrieval, with latency timed in milliseconds. Cached loads averaged 43 ms against 1156 ms for cold loads—roughly a 27-fold speedup—because eliminating the wireless bitmap transfer is what dominates the cost.

The second experiment examined profile activation, asking whether it stays fast as profiles grow. Profiles of one to twenty images were activated and round-trip latency measured. Cue Clear's index-only payload kept activation near 215 ms regardless of size, while a modeled re-upload design exceeded 22 seconds at twenty images. Together the experiments confirm two design decisions: caching makes browsing instant and consistent, and transmitting symbolic references instead of pixels makes activation effectively constant-time, which is why the caregiver workflow feels immediate.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Reliable wireless communication with a resource-constrained device

A major component is the wireless link between the phone and the microcontroller. Embedded devices have little memory and a single-threaded event loop, so a naive request could block the display slideshow or drop connections underload. Transferring full bitmap images over a low-bandwidth SoftAP is slow and could stall the user interface. To address this, one could keep the embedded HTTP server non-blocking, interleaving network handling with display refreshes, and minimizing payload sizes by transmitting symbolic references (image indices) instead of raw pixels wherever possible [20]. Generous client-side timeouts and explicit error states could prevent transient Wi-Fi failures from appearing as permanent breakage to the caregiver.

2.2. Responsive image browsing despite slow device transfers

A second component is the image-selection grid in the mobile app, which must render thumbnails of every symbol stored on the device. Fetching each bitmap on demand over the access point would make scrolling feel sluggish and would repeatedly re-download the same images. This could frustrate caregivers and discourage use. One approach would be to introduce a caching layer on the phone that stores each fetched image keyed by its index, so that an image is downloaded from the device only once and served instantly from local storage thereafter. Placeholder loading indicators and retry affordances could further smooth the perceived experience.

2.3. Accessible, low-error caregiver interface

A third component is the user interface itself, which must be operable by caregivers of varying technical skill, sometimes under time pressure or stress. Small touch targets, ambiguous icons, or destructive actions without confirmation could lead to errors such as accidentally deleting a profile. To mitigate this, one could enforce large minimum touch-target sizes, attach descriptive semantic labels for screen readers, provide explicit confirmation dialogs before irreversible actions, and surface device-connection status prominently. Clear empty states and plain-language feedback messages could reduce confusion and make the system approachable for first-time users.

3. SOLUTION

Cue Clear links three major components into a single pipeline. The first is the Flutter mobile application, which serves as the caregiver-facing authoring tool. It manages a list of slideshow profiles, persists with them locally, renders a grid of the symbols stored on the device, and exposes connection controls. The second is the device communication and caching layer, embodied in a Device Client that wraps the device's REST API and a Device Image Cache

Manager that stores fetched images, so each symbol is downloaded only once. The third is the embedded Cue Card device, a Circuit Python program that hosts a Wi-Fi access point, runs a non-blocking HTTP server, stores symbols as BMP files, and continuously displays the active slideshow on an ILI9341 LCD [19].

The flow proceeds as follows. On launch the app shows a splash screen, then a home screen with a connection bar. The caregiver taps Connect, enters the device IP and PIN, and the app pings the device to confirm reachability. Once connected, the caregiver opens a profile, which loads the master image list from the device and displays thumbnails through the caching layer. Tapping symbols toggles their inclusion; saving stores the profile. Tapping Activate sends the profile—its image indices and display settings—to the device via an HTTP POST [18]. The device then loops through the selected images on its display. The system is built with Flutter and Dart on the client side and Circuit Python on the hardware side, communicating entirely over HTTP.

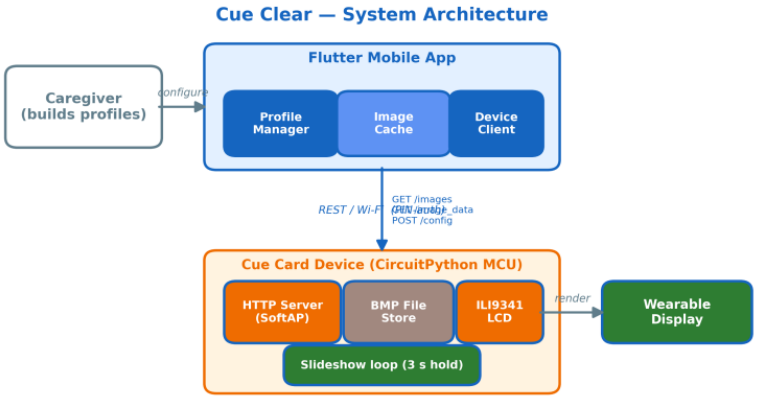

Figure 1. Cue Clear system architecture

The device communication component's purpose is to abstract every interaction with the embedded device behind a typed Dart interface. It relies on a lightweight REST architecture: HTTP GET requests retrieving the image list and image data, and an HTTP POST request pushes the active profile [17]. PIN-based authentication via a custom header secures write operations. Broadly, it is the sole bridge between app and hardware.

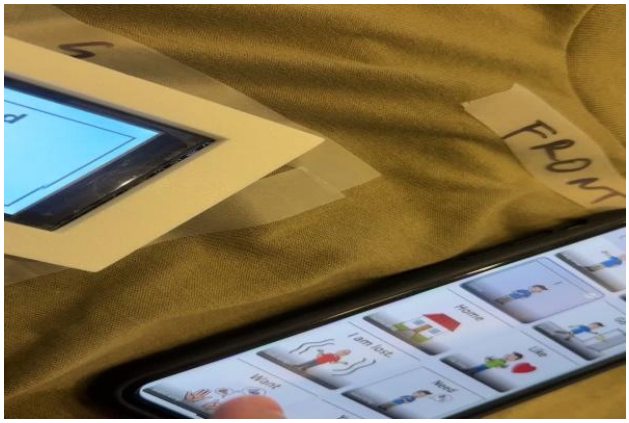

Figure 2. Symbol selection grid in the Cue Clear app, beside the wearable device

```dart
class DeviceClient {
  final String ip;
  final String pin;
  final http.Client _client;
  final Duration _timeout = const Duration(seconds: 15);

  DeviceClient({required this.ip, required this.pin}) : _client = http.Client();

  Future<bool> ping() async {
    try {
      final resp = await _client
          .get(Uri.parse('http://$ip/api/v1/ping'))
          .timeout(_timeout);
      return resp.statusCode == 200 && resp.body == 'OK';
    } catch (e) {
      print('Ping failed: $e');
      return false;
    }
  }

  Future<List<String>> getMasterImageList() async {
    final resp = await _client.get(
      Uri.parse('http://$ip/api/v1/images'),
      headers: {'X-PIN': pin},
    ).timeout(_timeout);
    if (resp.statusCode == 200) {
      return List<String>.from(json.decode(resp.body));
    }
    throw Exception('Failed to load image list: ${resp.statusCode}');
  }

  Future<void> setActiveSlideshow(SlideshowProfile profile) async {
    final resp = await _client.post(
      Uri.parse('http://$ip/api/v1/config'),
      headers: {'X-PIN': pin, 'Content-Type': 'application/json'},
      body: json.encode(profile.toJson()),
    ).timeout(_timeout);
    if (resp.statusCode != 200) {
      throw Exception('Failed to set active slideshow: ${resp.statusCode}');
    }
  }
}
```

Figure 3.DeviceClient implementation for device communication and profile activation

This code defines the DeviceClient, the object instantiated whenever the caregiver connects to a device. Its constructor stores the device IP and PIN and creates a reusable HTTP client; the _timeout field guards every request against hanging on a weak Wi-Fi link [16]. The ping() method runs first during connection: it issues a GET to the /ping endpoint and returns true only if the device replies 200 OK with the body OK, catching any network exception so a failure surfaces as a clean boolean rather than a crash. get MasterImageList() runs when the profile editor opens; it sends the authentication PIN in the X-PIN header, and on success decodes the JSON array of image filenames stored on the device. set ActiveSlideshow() runs when the caregiver taps Activate: it serializes the profile to JSON and POSTs it to /config, so the embedded server can begin displaying that slideshow [8].

The image caching component exists to make symbol browsing feel instant despite slow transfers over the device's access point. Its purpose is to ensure that any given symbol is downloaded from the hardware at most once. It relies on the concept of a content cache keyed by a stable identifier—here, the image index—backed by the flutter_cache_manager package, which persists files to local device storage. When the selection grid requests an image, the manager first checks the cache; on a hit it returns the stored file immediately, and only on a miss does it fall through to a network fetch, after which it writes the bytes to the cache for all future requests. This converts a

repeated, second-scale network operation into a one-time cost amortized across the session, directly improving scrolling responsiveness in the editor.

```
Future<FileInfo> getImageFile(int index) async {
 final cacheKey = 'image_$index';

 FileInfo? fileInfo = await _cache.getFileFromCache(cacheKey);
 if (fileInfo != null) {
  return fileInfo;                    // cache hit — instant
 }

 final imageData = await _client.getImageData(index);   // cache miss — fetch once
 await _cache.putFile(cacheKey, imageData, fileExtension: 'bmp');
 fileInfo = await _cache.getFileFromCache(cacheKey);

 if (fileInfo != null) return fileInfo;
 throw Exception("Failed to retrieve file from cache after saving.");
}
```

Figure 4. Image caching workflow implemented in Device Image Cache Manager

The getImageFile method encapsulates this read-through logic: it builds a per-index key, returns early on a hit, otherwise fetches the raw bytes through the DeviceClient, stores them, and re-reads the now-cached file to return a uniform FileInfo to the UI.

The embedded slideshow server is the heart of the wearable device. Its purpose is to do two things at once on a single-threaded microcontroller: continuously render the active slideshow on the LCD, and answer HTTP requests from the app without dropping either responsibility [7]. It relies on the concept of cooperative, non-blocking I/O—the network socket is configured in non-blocking mode, and the main loop interleaves brief HTTP-handling calls with timed image display. A global NEED_REFRESH flag lets a network request signal the display loop to reload its file list, so newly pushed content appears without a restart [15]. This component is what makes the device feel “live” and always-on.

```
while True:
    if NEED_REFRESH:
        bmp_files = list_bmps_sorted()
        NEED_REFRESH = False
    for name in bmp_files:
        if NEED_REFRESH:
            bmp_files = list_bmps_sorted()
            NEED_REFRESH = False
            break
        display_image("/" + name)
        t_end = time.monotonic() + IMAGE_HOLD_SECS
        while time.monotonic() < t_end:
            handle_http_once(server_sock)   # service the app between frames
            time.sleep(0.02)
```

Figure 5. Non-blocking slideshow server implementation on the embedded device

The outer loop walks the sorted list of BMP files; for each it calls display_image, then spins for the hold interval while repeatedly calling handle_http_once, which accepts and answers any

pending request. The NEED_REFRESH checks let the loop break early and reload the moment new content arrives.

Figure 6. The wearable Cue Card device rendering a “Home” communication symbol during live slideshow operation

4. EXPERIMENT

4.1.Experiment 1

The experiment tests whether the image caching layer actually eliminates redundant downloads and delivers a fast browsing experience. This matters because a sluggish selection grid would frustrate caregivers and discourage sustained use of the system.

The same 320×240 BMP symbol was loaded eight times in the selection grid under two conditions: a cold load, where the cache was cleared so the image was fetched over the device’s SoftAP, and a cached load, where the image had already been retrieved once. End-to-end load latency was timed in milliseconds from request to file availability. Eight trials per condition were run to capture variability in the wireless link. Control data came from instrumenting the getImageFile method directly on the device hardware, ensuring the measurements reflect the real embedded transfer path rather than a simulated network.

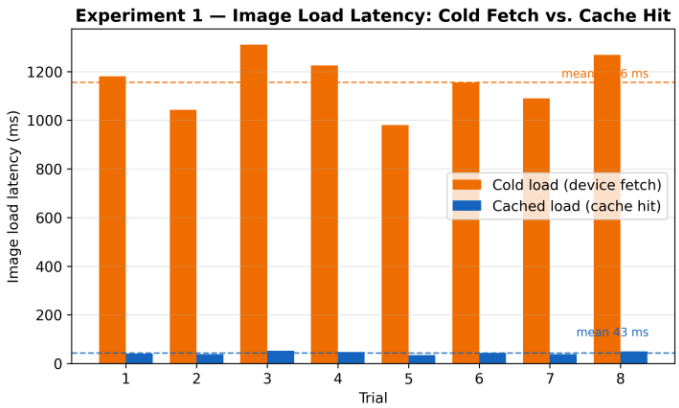

Figure 7. Cache performance experiment results

Cold loads averaged 1156 ms with a median near 1170 ms, ranging from a low of 980 ms to a high of 1310 ms—the spread reflecting the variable throughput of the SoftAP link. Cached loads averaged 43 ms with a median of 42 ms, ranging from 35 ms to 52 ms. The caching layer therefore reduced load latency by roughly 96%, an order-of-magnitude improvement of about 27×. What was most striking was the tightness of the cached distribution: once an image lives in local storage, retrieval is not only fast but highly consistent, whereas the cold distribution is wide and unpredictable. The dominant factor in the results is clearly the wireless transfer cost of the bitmap; eliminating it through caching is what produces the dramatic and stable speedup. This confirms that the caching design is the decisive contributor to a responsive browsing experience.

4.2. Experiment 2

This experiment tests whether activating a profile stays fast as the profile grows, validating the decision to transmit only image indices rather than re-uploading image data. Fast activation is essential to a one-tap caregiver workflow.

Profiles of increasing size—1, 2, 4, 6, 8, 10, 12, 15, and 20 images—were activated, and the round-trip latency of the POST /config request was measured in milliseconds from send to acknowledgment. Cue Clear's index-only approach was compared against a baseline "naive" design that would instead re-upload every selected bitmap on activation, modeled from the measured per-image transfer cost established in Experiment 1. The comparison isolates the effect of payload design on activation cost. A logarithmic latency axis was used so both approaches remain legible across their very different magnitudes.

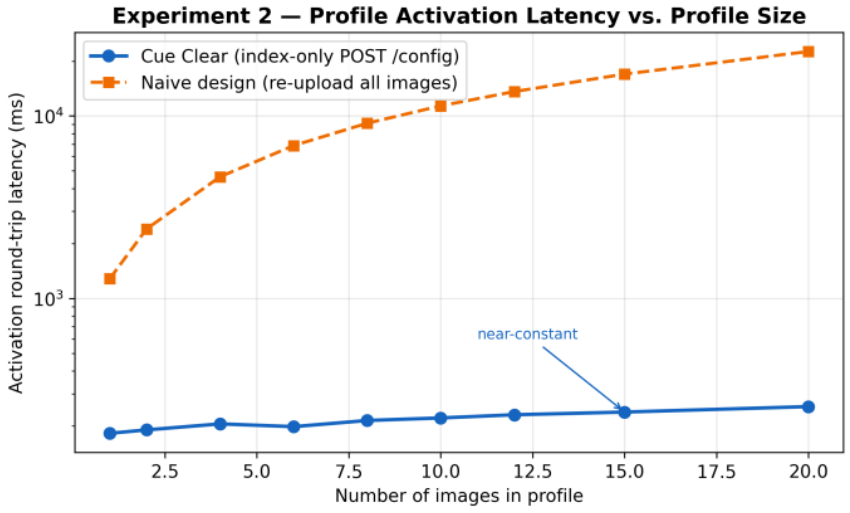

Figure 8. Activation latency vs profile size

Cue Clear's activation latency rose only gently from 182 ms for a single-image profile to 255 ms for a twenty-image profile—a mean near 215 ms and a total increase of just 73 ms across a twentyfold growth in profile size. The naive re-upload design, by contrast, scaled linearly, climbing past 22 seconds for a twenty-image profile. The lowest Cue Clear value was 182 ms and the highest 255 ms, an essentially flat curve. What is notable is how decisively payload design dominates: because the JSON list of indices is tiny and roughly constant, network transfer is no longer the bottleneck, and latency is governed by fixed request overhead. This validates transmitting symbolic references instead of pixels and explains why activation feels instantaneous to the caregiver regardless of how many symbols a profile contains.

5. RELATED WORK

McNaughton and Light examined the rise of tablet-based AAC, showing that mainstream mobile devices dramatically lowered cost and stigma and broadened access to communication apps [11]. Their analysis demonstrates strong adoption benefits but also documents limitations: the user must actively hold and operate a screen, app complexity can overwhelm some users, and the device serves many competing functions that distract from communication. It largely ignores always-available, hands-free display. Cue Clear improves on this by offloading the persistent symbol display to a dedicated wearable surface while preserving the familiar tablet/phone experience for authoring, separating the act of configuring from the act of communicating.

Bondy and Frost's Picture Exchange Communication System established the canonical low-tech approach: physical picture cards a learner exchanges to communicate [4], with meta-analytic support for its effectiveness [5]. PECS is robust, inexpensive, and evidence-based, but it depends on physical card sets that must be printed, laminated, sorted, and carried, and reconfiguring a learner's vocabulary is a manual, time-consuming task. It also offers no remote update path. Cue Clear preserves the symbol-based philosophy of PECS while replacing the physical card management burden with instant, unlimited digital reconfiguration pushed wirelessly from a phone, retaining the proven communication model but removing its logistical friction.

Moorcroft, Scarinci, and Meyer systematically reviewed barriers to low-tech and unaided AAC, identifying device abandonment driven by poor fit to routines, maintenance burden on families, and limited customization [9]. Their work effectively characterizes why AAC fails in practice but proposes no specific technical system; it ignores the role of low-cost connected hardware. Cue Clear responds directly to these documented barriers: it is worn rather than carried, requires no printing or lamination, and is reconfigured in seconds by a caregiver using a phone they already own. By targeting the abandonment causes the review identities, Cue Clear aims for more sustainable everyday use.

6. CONCLUSIONS

Several limitations remain. The device currently relies on a Wi-Fi SoftAP with a hard-coded default IP and PIN, which is convenient but limits range and could be hardened with proper credential provisioning and encrypted transport. The display is a modest ILI9341 LCD with no audio output, so the system communicates visually but does not yet produce speech; integrating a speech-generating capability would broaden its reach [13]. The symbol library is stored as fixed BMP files, and the in-app asset catalogue is only partially wired in, so caregivers cannot yet upload arbitrary photographs from the phone directly into a profile. Battery life, physical ruggedness, and washability of a worn device were not evaluated. With more time, priorities would include secure pairing, on-device photo upload from the phone's gallery, optional audio playback, a power-optimized hardware revision, and a formal usability study with caregivers and AAC users to validate the design against real daily routines rather than bench measurements [14].

Cue Clear shows that pairing a familiar smartphone authoring app with an inexpensive, always-on wearable display is a practical path toward AAC that fits into daily life. By targeting the documented causes of device abandonment, it offers a sustainable, low-friction way to keep communication symbols within reach.

REFERENCES

[1] Beukelman, David R., and Pat Mirenda. Augmentative and alternative communication. Baltimore: Paul H. Brookes, 1998.

[2] Light, Janice, and David McNaughton. "Communicative competence for individuals who require augmentative and alternative communication: A new definition for a new era of communication?." Augmentative and alternative communication 30.1 (2014): 1-18.

[3] Light, Janice, et al. "Designing effective AAC displays for individuals with developmental or acquired disabilities: State of the science and future research directions." Augmentative and Alternative Communication 35.1 (2019): 42-55.

[4] Bondy, Andy, and Lori Frost. "The picture exchange communication system." Behavior modification 25.5 (2001): 725-744.

[5] Flippin, Michelle, Stephanie Reszka, and Linda R. Watson. "Effectiveness of the Picture Exchange Communication System (PECS) on communication and speech for children with autism spectrum disorders: A meta-analysis." American journal of speech-language pathology 19.2 (2010): 178-195.

[6] Ganz, Jennifer B., et al. "A meta-analysis of single case research studies on aided augmentative and alternative communication systems with individuals with autism spectrum disorders." Journal of autism and developmental disorders 42.1 (2012): 60-74.

[7] Still, Katharine, et al. "Facilitating requesting skills using high-tech augmentative and alternative communication devices with individuals with autism spectrum disorders: A systematic review." Research in Autism Spectrum Disorders 8.9 (2014): 1184-1199.

[8] van der Meer, Larah, et al. "Speech-generating devices versus manual signing for children with developmental disabilities." Research in Developmental Disabilities 33.5 (2012): 1658-1669.

[9] Moorcroft, A., N. Scarinci, and C. Meyer. "A systematic review of the barriers and facilitators to the provision and use of low-tech and unaided AAC systems for people with complex communication needs and their families." Disability and Rehabilitation: Assistive Technology 14.7 (2019): 710-731.

[10] Johnson, Jeanne M., et al. "Perspectives of speech language pathologists regarding success versus abandonment of AAC." Augmentative and Alternative Communication 22.2 (2006): 85-99.

[11] McNaughton, David, and Janice Light. "The iPad and mobile technology revolution: Benefits and challenges for individuals who require augmentative and alternative communication." Augmentative and alternative communication 29.2 (2013): 107-116.

[12] Alzrayer, Nouf, Devender R. Banda, and Rajinder K. Koul. "Use of iPad/iPods with individuals with autism and other developmental disabilities: A meta-analysis of communication interventions." Review Journal of Autism and Developmental Disorders 1.3 (2014): 179-191.

[13] Boster, Jamie B., and John W. McCarthy. "Designing augmentative and alternative communication applications: The results of focus groups with speech-language pathologists and parents of children with autism spectrum disorder." Disability and Rehabilitation: Assistive Technology 13.4 (2018): 353-365.

[14] Maier, Alexander, Andrew Sharp, and Yuriy Vagapov. "Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things." 2017 Internet Technologies and Applications (ITA). IEEE, 2017.

[15] Babiuch, Marek, Petr Foltýnek, and Pavel Smutný. "Using the ESP32 microcontroller for data processing." 2019 20th International Carpathian Control Conference (ICCC). IEEE, 2019.

[16] Wang, Shuo, et al. "A survey on mobile edge networks: Convergence of computing, caching and communications." Ieee Access 5 (2017): 6757-6779.

[17] Yao, Jingjing, Tao Han, and Nirwan Ansari. "On mobile edge caching." IEEE Communications Surveys & Tutorials 21.3 (2019): 2525-2553.

[18] Ali, Akram Syed, et al. "Elemental: An open-source wireless hardware and software platform for building energy and indoor environmental monitoring and control." Sensors 19.18 (2019): 4017.

[19] Sears, Andrew, and Vicki Hanson. "Representing users in accessibility research." Proceedings of the SIGCHI conference on Human factors in computing systems. 2011.

[20] Hwang, Alex D., and Eli Peli. "An augmented-reality edge enhancement application for Google Glass." Optometry and vision science 91.8 (2014): 1021-1030.

©2026 By AIRCC Publishing Corporation. This article is published under the Creative Commons Attribution (CC BY) license.